DEVELOPMENT AND PROTOTYPING OF HAPTIC INTERACTIONS FOR DATA EXPLORATION

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT AT CANTERBURY
IN THE SUBJECT OF COMPUTER SCIENCE
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

By
Sabrina A. Panëels
January 2010

F221059

Acknowledgements

I would like to thank my supervisors, Jonathan Roberts and Peter Rodgers, for their advice, guidance and support, throughout the last four years. This PhD would have not been possible without their help but also without the Computing Laboratory of the University of Kent, that gave me the opportunity to do a PhD by funding it as well as many travels to interesting conferences and the opportunity to meet many people who have helped me along the way.

I thank my parents, for their love, advice and support throughout the PhD as well as their patience, especially in the tough times. Un grand merci à ma famille, qui m'a soutenue pendant toute la durée de ma thèse et surtout dans les moments difficiles, sans vous je ne serais pas arrivée là où j'en suis aujourd'hui.

I am indebted to Poul Henriksen, my best friend and my "third supervisor", without whom I would have not completed this PhD. He helped me face better the problems the PhD can be paved with and also gave me the opportunity to discuss my research and brainstorm ideas, even though our research areas were not related. Thank you for your patience and invaluable help and for everything else!

Finally, I wish to thank all my friends and colleagues, both from UK and France: the 3WW team, Benoit Dillet, Tara Puri, Edward Suvanaphen (the Canterbury team), Srivas Chennu, Patrick Craston and Katherine Johnson; colleagues and friends from the University, Phil Cattani, Lingfang Du, Sébastien Marion, Axel Simon and Rick Walker; my only haptic buddy in the UK Frank Vidal; and all my friends from France, especially Phuong Vo N'Guyen, Isabelle Vivier, Cyrielle Guignard and les Zelist. And also thank you to all the other friends I have not cited.

To summarize my thesis, a famous quote: "Touch touch".

Abstract

Haptic technology means the sense of touch can be added to computer applications. Not only can the user see and hear information, but the user can also feel it! Thus, haptics provides new, exciting ways of conveying and interacting with information.

In particular, haptics can be used to reveal the underlying data of a representation, often referred to as haptic (data) visualization, to augment available modalities (visual, auditory) or as an alternative in cases where other modalities are impractical, overloaded or not available, such as for visually impaired people. Haptic visualization is a recent area, not yet fully established, where considerable research needs to be conducted, in particular concerning effective representations and interactions to convey information. Haptic technology presents several limitations, in particular, the bandwidth of haptics is smaller than vision or audio, hence, designing effective haptic visualizations is very challenging. Therefore, this thesis first comprehensively reviews the work achieved so far in haptic data visualization and presents a solution to the open haptic overview challenge, both for a line chart application, through the combination of guidance metaphors and free exploration and a scatter plot using a force model.

While investigating the haptic visualization area and developing the line chart application, many challenges were encountered that slowed down the testing of new interaction ideas. Developing and testing new haptic ideas is not a simple task and can be time-consuming: the various haptic APIs still require the user to have good programming skills, a good understanding of haptic interactions and technical knowledge of the devices to be used. Consequently, methods to facilitate the development of haptic applications and in particular haptic interactions, crucial to haptic visualization, are needed. Therefore, these methods were investigated and led to the implementation and evaluation of a prototyping tool, HITPROTO, that aims to enable the rapid prototyping of haptic interactions by people with no programming skills.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The word "haptics" comes from the Greek word haptikos relating to the sense of touch [FFM$^+$04]. Srinivasan [Sri04] defines haptics as referring to "sensing and manipulation through touch". Haptics includes two major categories of sensory understanding: tactile and kinesthetic [Sri04]. Tactile sensations include the sensation of shape, textures and vibrations, while kinesthetic sensations represent physical forces, frictional resistance and weight [YJN$^+$96, SB97, BS02b, FFM$^+$04, Cas05]. Physiological and perceptual details about the haptic sense can be found in Chapter 6 of [YJN$^+$96], and in [BS02b, FFM$^+$04, Cas05].

Haptic technologies have been around for the last 50 years. However, the development and application of haptics experienced a breakthrough in 1993 [FFM$^+$04] when Massie and Salisbury [MS94] developed the "PHANTOM Haptic Interface", which was made commercially available the same year by SensAble^{TM} Technologies [Sen10]. Since then, other effective haptic devices have been developed and commercialized, using a wide range of technologies [Cas05, YJN$^+$96]. These devices have been used by researchers in various areas, including medicine, virtual reality, accessibility or gaming [FFM$^+$04, HACH$^+$04, BS02b]. The process of haptic perception with a haptic interface includes three main steps [Sri04, YJN$^+$96]. Firstly, the device senses the user's position. Secondly, the forces or vibrations to be rendered on the device's actuators are computed by the haptic model (collision detection and response). Thirdly, these effects are applied to the user through the device [Sri04, Cas05].

Although both tactile and kinesthetic sensations are combined in the human sensory system, creating devices providing both sensations is difficult and the devices have evolved separately, at different paces according to the type of haptic sensations [FFM$^+$04].

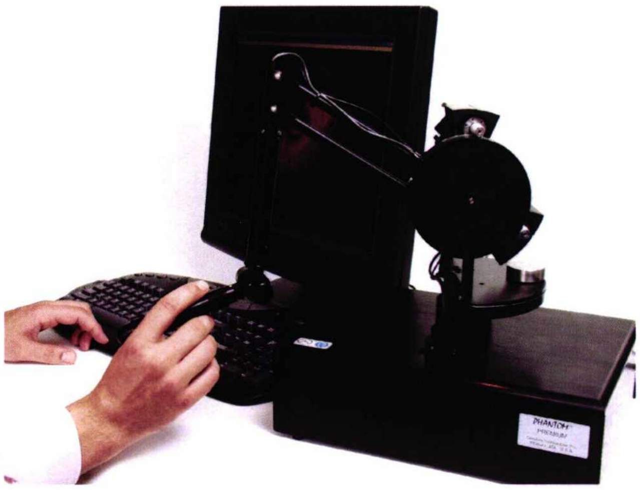


Figure 1: The PHANTOM device, by SensAble technologies

Force Feedback Technology

Force feedback research emerged with the developments of teleoperation and robotics [YJN+96, FFM+04]. Defined by Stone [Sto00] as the "extension of a person's sensing and manipulation capability to a remote location", teleoperation was used to access dangerous areas, such as in nuclear sites or subsea fields, for manipulation and maintenance where force reflection was integrated into manipulators to increase precision. Later effort focused on integrating force feedback in Virtual Environments (VE) for similar reasons. The most famous example of manipulator application in VE is the GROPE project [BJOYBK90], where force feedback was used for molecular docking. The user could view the molecules, interact and feel the different forces from individual molecules using the force-feedback manipulator.

The development and application of force-feedback devices experienced a breakthrough after Massie and Salisbury [MS94] developed and commercialized the "PHANTOM haptic interface" (see Figure 1).

Since then, many other devices have been developed. Examples include datagloves with integrated pneumatic actuators [BRS+92, BZR+92, BL93] or a nine degree-of-freedom master manipulator consisting of three sets of pantograph link mechanisms and coaxial actuators [Iwa90] (see Figure 2). Many technologies have been explored to build these devices, with the goal of providing realistic and intuitive haptic perception. A list of these technologies can be found in [Cas05] and advantages and disadvantages are analyzed in Chapter 6 of [YJN+96], as well as

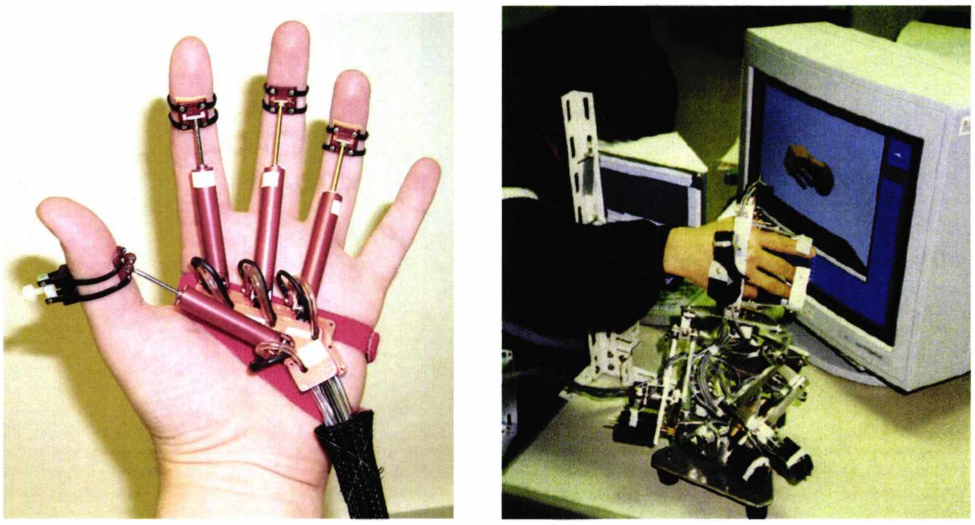

(a) The Portable Dextrous Master [BZR+92] (b) 9 DOF Master Manipulator [Iwa90]

Figure 2: Examples of Masters with force feedback

in "Designing with Force-Feedback" in [FFM+04].

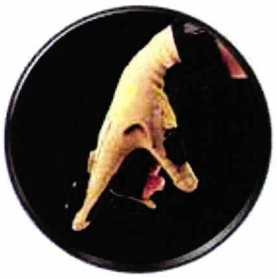
Tactile Feedback Technology

The research of tactile feedback methods has been driven by various motivations: the motivation to increase accessibility for visually impaired people (i.e., to investigate better and more intuitive ways to interact with information), to enhance current interfaces (especially on mobile and small devices) and to study the tactile sense.

Even with current technology, achieving realistic tactile feedback is a difficult task. Various devices have been used to provide tactile feedback (see Figure 3), such as gloves with piezoceramic benders [ZLB+87], solenoid with plungers [Pet67] and pin arrays (see Figure 3(b)). Benali-Khoudja et al. [BKHAK04] give a comprehensive list of existing devices, while the technologies and their advantages and disadvantages are described in Chapter 6 of [YJN+96], as well as in [Cas05].

Technologies Today

The research in both tactile and force feedback methods has enabled the development of efficient devices, and as a consequence, interest in haptics has grown. Today, there are many technologies available, both in the research domain and


(a) CyberTouch

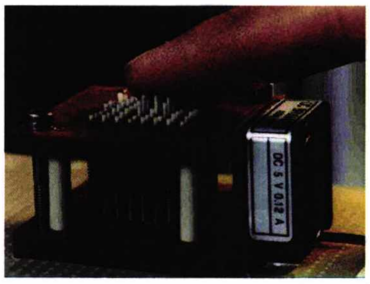
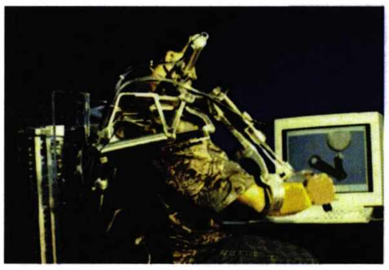
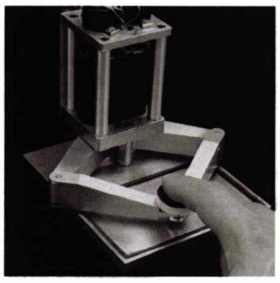

(b) Tactile Display

Figure 3: Examples of devices for tactile feedback: a) the CyberTouch from Immersion Corporation and b) a Tactile Display developed at the Forschungszentrum Karlsruhe.


(a) Pneumatic exoskeleton


(b) Desktop Pantograph

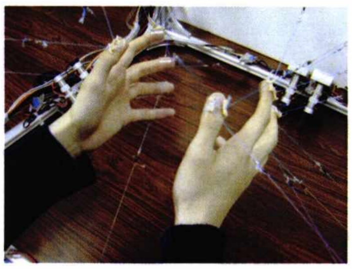

(c) SPIDAR-8

Figure 4: Tactile and Force Feedback devices: a) a pneumatic exoskeleton by Southern Methodist University, b) the desktop Pantograph from McGill University and c) SPIDAR-8 by the Tokyo Institute of Technology.

available to purchase; pictures and details of many haptic tools can be seen in the 'Haptics Photo Gallery' [Hap04] (and in [YJN+96, BS02b, Cas05, Har09, FFM+04]), and in the review papers by Hayward et al. [HACH+04] and Laycock and Day [LD03]. The devices (see Figure 4) include gloves with tactile or force feedback [BRS+92, BL93, BZR+92] (see Figures 3(a) and 2(a)), exoskeletons providing feedback to the hand [Iwa90] or the whole arm (see Figure 4(a)), space balls and arms, joysticks, mice and wheels. Other interfaces use wires (see Figure 4(c)), pens (see Figure 1), or pin arrays (see Figure 3(b)).

There has been a definite progression from 'infant' haptic devices through teleoperation to today, where haptic devices are used to interact with computer applications. The advent of haptics has enabled the emergence and growth of the haptic data visualization area, also referred to as 'haptification', where new ways of presenting information non-visually have been investigated.

Visualization Applied to Haptics

The term "Visualization" is commonly used to mean 'a graphical realization' of some data. But, visualization is much more than merely the representation of data through computer graphics: it is the understanding of information through any modality whether visual, auditory, olfactory or indeed haptics [Owe99]. Hence, Haptic Data Visualization (HDV) can be defined as the understanding of information through tactile and kinesthetic senses.

The specific aim of HDV is to give the user understanding of the *data* that is being represented by the haptic model; not only does the user feel the model, but in doing so realizes *value* and can draw conclusions from that data. For instance, a model representing a line graph of stock market data would aid the user to understand how the stock values change over time, whether increasing or decreasing. A user may also be able to perceive maximum or minimum values or points of crossover or inflection on the graph. Values may be realized through different ways, e.g. larger values could be mapped to high frequencies, with low values to low frequency vibrations.

It is useful to consider the HDV process in three parts: model, rendering and device. Developers must carefully consider each part to present the data effectively to the user. Effective designs for HDV thus consider the whole process together. The principal differences with HDV and other uses of haptics is specifically the makeup of the model.

Creating an effective model is a principal component of HDV. The developer needs to decide what data is going to be presented; how this information will be mapped and transformed into haptic properties so as to determine how the user will perceive *value*; and how the user is to navigate. This process is similar to that of the dataflow paradigm of traditional visualization [HM90], see Figure 5. But HDV presents particular challenges to developers. First, the data needs to be processed, simplified and often idealized. For instance, it may be that the user is interested in overall trends, in which case averages over some of the data would be more useful than the original collected values. Second, the information is then mapped to various haptic variables to build the haptic model of this information. However, there are many haptic variables that can be used to perceive information including actuator position, force-strength, vibration frequency, and surface texture. It is not often clear to a developer which mappings should be used, and what the limitations of each variable are (the limitations of each variable is also specific to the device being used). Developers also need to decide how the user

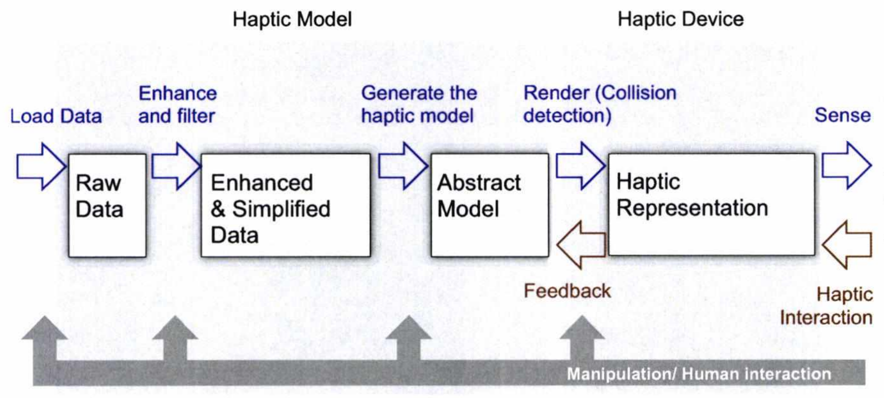


Figure 5: Haptic Visualization Process

is going to interact with the information; many haptic interfaces rely on the user being active in the environment to perceive the information. Furthermore, users can potentially modify any parameters to effect change in the system such as altering the filtering, mapping or viewpoint of what is being represented. This enables the haptic visualization to be dynamic, which is not possible with many technologies that enable static haptic displays (often used to feel graphs etc.) such as swell paper or bread-board configurations using pins and bands. Subsequently, the haptic model is rendered in terms of forces and/or vibrations the user can feel.

Haptic rendering is utilized to generate the desired presentation. More details about haptic rendering can be found in the book chapter by Basdogan and Srinivasan [BS02a], the book by Lin and Otaduy [LO08] and the survey by Laycock and Day [LD07]. However, it broadly consists of two processes: collision detection and collision response. In the particular case of a force-feedback device such as the PHANTOM^{TM}, being an input and output device, if the user touches a solid wall from the haptic model the collision-response mechanism transmits the right force to the device to mimic a solid surface. Thus, when the user tries to push through the solid wall the device transmits a greater force. A useful survey on collision detection is by Lin and Gottschalk [LG98], and although published ten years ago, it provides a good overview of the main collision detection algorithms.

1.1 Motivation

As discussed above, haptic visualization is a growing research area. However, the use of haptic interactions for visualization is not widespread [PR09, RP07] when compared to the integration of haptics in the medical area or teleoperation. When designing a new haptic application, developers encounter many challenges such as filtering the data and finding appropriate and effective mappings and metaphors both for the representation and user interactions. The design of the mapping and metaphors is not necessarily obvious, consequently developers need to ideate and test new designs. In particular, many challenges remain unsolved in conveying information most efficiently while overcoming the limitations of haptic devices (e.g. tactile or force feedback only, size of the workspace, resolution, point-based). Designs to provide a solution to the difficulty of providing a haptic overview with point-based force feedback devices are explored in this thesis for haptic charts, in particular line charts and scatter plots.

In addition to facing challenges with the design of metaphors for a given application, developing haptic applications and interactions is still difficult and time-consuming, and although various APIs are available (which provide a generic interface to multiple devices), they still require the user to have good programming skills, a good understanding of haptic interactions and technical knowledge of the devices to be used. Guidelines and methodologies have been proposed to facilitate the development of haptic applications, but they fall short when it comes to speeding the programming process and therefore facilitating the rapid testing of new designs. Consequently, there is a need for prototyping tools that can be used to quickly implement and test haptic interactions. The demand for such toolkits has long been identified and satisfied in related research areas, such as virtual reality, where industry has access to CAD software and, more generally, there are user-oriented authoring tools to create virtual worlds. The ability of designers to quickly design and test interaction metaphors enables the creation of an enriched experience for users of the virtual world. In the context of data visualization, rapid development and testing should encourage the development and exploration of new haptic interactions, as well as permitting a wider audience to explore the possibilities of haptic visualization and therefore foster the area. Prototyping tools have been developed to deal with multisensory applications and the increasing range of "Non-WIMP" devices, but most do not clearly support haptic devices, therefore the development of such a tool is also investigated in this thesis.

Therefore, the motivations can be summarized as follows:

- Haptic interactions are not widespread for visualization and need to be further explored to foster the haptic visualization area.
- The design of a haptic visualization involves many challenges, such as finding

effective mappings and metaphors, both for the representation and user interactions. Such a challenge that needs to be addressed concerns the haptic overview, as getting an overview is an important step in visualization to understand the information.

- The rapid development and testing of haptic interactions is difficult and time-consuming, which underlines the need for prototyping tools. However, most tools developed for the prototyping of multisensory interactions do not clearly support haptics, which emphasizes the need to investigate a solution for the prototyping of haptic interactions.

1.2 Contributions and Publications

This thesis provides three main contributions which are listed below.

Comprehensive Review of Designs for Haptic Data Visualization

As explained above and in the paper "Where are We with Haptic Visualization" [RP07], haptic visualization is a recent area. As a matter of fact, even the term to refer to this area is not yet established: some people use the term haptification or haptization rather than haptic visualization. However, these terms have also been used to mean the mere "addition" of haptics to an application. As it is a relatively infant area, a review of existing work helps collate the different designs and ideas, draw conclusions about the research conducted so far, identify gaps or unsolved challenges as well as draw inspiration for future work. In the haptic application area, a few reviews are available and often categorize the research by areas, such as teleoperation, gaming, virtual reality and accessibility. In the case of haptic visualization though, only some of the work presented in previous reviews qualifies as "haptic visualizations". As no reviews was found for this area, the first contribution was to compile a comprehensive review of the designs for the haptic visualization area that is based on a classification which categorizes the designs on the type of representation (Charts, Maps, Signs, Networks, Diagrams, Images & Photo-realistic Renderings and Tables) rather than the area. This classification was developed as designs and user interactions are often re-used across areas but tend to be more consistent across representations, which share similar data and therefore similar goals. For instance, in charts, important features to be conveyed often include maximum and minimum points, general trends, thus leading the haptic design in a certain direction, while for images, all the features are equally important leading to a different strategy for the design. This review was first published in the article entitled "Review of Designs for Haptic Data Visualization" [PR09] and is further expanded in Chapter 2.

[RP07] Jonathan C. Roberts and Sabrina Panëels. Where are we with Haptic Visualization? In *WorldHaptics - Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC '07)*, pages 316-323, Tsukuba, Japan, March 2007. IEEE Computer Society Press.

[PR09] Sabrina Panëels and Jonathan C. Roberts. Review of Designs for Haptic Data Visualization. *Transactions on Haptics*, 2009. PrePrint (to appear in the April-June 2010 issue).

New Designs for Haptic Charts

Haptic line charts have already been investigated, to make them accessible to visually impaired people. However, many challenges remain, such as conveying the overview and important features (especially for point-based devices) and representing and discriminating several lines. Conveying an overview haptically is a general challenge that does not only apply to chart techniques, however within the scope of this thesis, this challenge was explored for charts, in particular for line charts. Previous work conveyed the overview of line charts using the audio modality and also dealt with representation on the "positive" axis only. The application presented in this thesis provides a haptic alternative and uses the whole display area to represent the line chart (therefore also dealing with intersections with axes). The haptic alternative consists in the implementation of guidance metaphors, originating from ideas described by Roberts et al. [RFC02]. These implementations are described in chapters 3 and 6 and also presented as a poster [PR07] and later in a paper [PRR09] at the Workshop for Haptic and Audio Interaction Design (HAID). An overview technique was also investigated for scatter plots, for which there has been little research in the haptic modality. The proposed method is described in Chapter 6 and in [PRR09].

[PR07] Sabrina Panëels and Jonathan C. Roberts. Haptic Guided Visualization of Line Graphs: Pilot Study. In Lorna Brown and Tae-Jeong Jang, editors, *Workshop on Haptic and Audio Interaction Design (HAID '07), poster and demo proceedings*, pages 5-6, Seoul, Korea, November 2007.

[PRR09] Sabrina Panëels, Jonathan C. Roberts, and Peter J. Rodgers. Haptic Interaction Techniques for Exploring Chart Data. In M. Ercan Altinsoy, Ute Jekosch, and Stephen Brewster, editors, *Workshop on Haptic and Audio Interaction Design (HAID '09)*, volume 5763 of *Lecture Notes in Computer Science*, pages 31-40, Dresden, Germany, September 2009. Springer.

Haptic Interaction Techniques Prototyping Tool

Finally, the third contribution of this thesis concerns the need for a prototyping tool to facilitate the design of haptic interactions, and in particular, make them accessible to a wider audience, including designers. The contribution is two-fold and includes the development and implementation of a visual prototyping tool for haptic interactions (HITPROTO) as well as its evaluation to assess whether people with no programming knowledge could effectively use the tool. HITPROTO aims to allow developers with no or little programming skills, such as blind students' teachers or designers, to explore interactions and more generally to stimulate the development of interactions to access data haptically. The evaluation was conducted to ensure that the designed visual language could achieve the main goal of rapid development of haptic interactions without requiring programming skills. The design of the tool as well as its evaluation through case studies was described in [PRR09], which forms the basis of Chapter 5 and part of Chapter 6. An in depth usability experiment with 9 participants is described in [PRR10], which forms the remainder of Chapter 6.

[PRR10] Sabrina Panëels, Jonathan C. Roberts, and Peter J. Rodgers. HITPROTO: a Tool for the Rapid Prototyping of Haptic Interactions for Haptic Data Visualization. In *Haptics Symposium (previously known as the Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems) (HAPTICS '10)*. IEEE Computer Society Press, 2010.

1.3 Summary of Thesis Structure

This chapter introduced the different concepts used throughout the thesis (haptics and haptic visualization) and presented the motivations and contributions of the research. The rest of the thesis is divided into three parts: 'Haptic Visualization Designs' which describes design solutions for certain types of representations; 'Facilitating the Development of Haptic Interactions' which explores methods to ease the development of haptic interactions, in particular for data visualization, and investigates the feasibility of the prototyping languages approach and 'Summary and Conclusions' which concludes the thesis. The chapters are organized as follows:

Part I

- **Chapter 2**: presents a review of designs for haptic data visualization classified according to the type of data representation (Charts, Maps, Signs, Networks, Diagrams, Images & Photo-realistic Renderings and Tables). In particular, it details the different ideas for mappings and interaction metaphors as well as results of evaluation, if applicable, for each category.
- **Chapter 3**: describes a haptic line chart application and the pilot study conducted to evaluate the guidance metaphors that were developed to convey the line chart overview.

Part II

- **Chapter 4**: gives related work around prototyping systems, in particular those that relate to haptic interactions.
- **Chapter 5**: covers the design and implementation of HITPROTO, a visual prototyping tool aimed at the rapid development of haptic interactions and accessible to non-programmers.
- **Chapter 6**: discusses the evaluation of the tool through the description of two case studies and the usability experiment conducted with 9 participants.

Part III

- **Chapter 7**: summarizes the achievements presented in the thesis and concludes with ideas for future work.

Part I

Haptic Visualization Designs

Chapter 2

Review of Designs for Haptic Visualization

Over the last two decades there has been a mounting interest in non-visual forms of presentation. Researchers have utilized touch and tactile devices, force-feedback joysticks, sound (sonification) and even smell (olfaction) to represent information [Lof03]. Not only can these methods be used to represent information to blind or partially sighted users, but they can also be useful in situations where the visual domain is either overloaded or impractical [Rob00].

In the area of Haptic Data Visualization (HDV), as explained in Chapter 1, the aim is to provide an understanding of the underlying data through effective data mappings and user interactions using the sense of touch. Rather than merely attempting to render the world, haptic feedback in visualization is used to convey information about the data being presented through well designed and efficient metaphors. The design of the metaphors and their haptic mapping is not necessarily obvious, consequently developers need to ideate new *designs* that enable effective HDV.

This chapter provides a comprehensive review and classification of designs for HDV, providing a snapshot of the state of the art. This chapter is an extended version of the review presented in the paper published under the same title [PR09].

Previous review papers have focused on specific aspects of haptics such as input devices [BKHAK04, Sto00, Lév05]; haptic rendering techniques [SB97, FB99, SCB04]; rendering in the medical domain [BDK$^+$04]. Other reviews have focused on the application of telepresence and Virtual Reality [YJN$^+$96, Sto00]. The work here is categorized by the representation of data: charts; maps; signs; networks;

diagrams; images and tables. This categorization was chosen because it distinguishes the various types of design, enabling the methods to be uniquely classified. E.g., the structure of a network presentation, that details associated data nodes, is very different to geographic information that is displayed on a map in two dimensions. For each different form, the design, the technologies used, and any issues or challenges with the presentation are described.

2.1 Classification

There are various possible taxonomies for HDV: those that classify based on the form of input data, the form of the output that is perceived, the sense they utilize, the task they perform or the application domain. Indeed, most of the previous review papers have classified the research by 'areas of application' [BKHAK04, Sto00, Lév05]. Bloomfield et al. [BDW$^+$03] on the other hand classified the techniques by tasks, in fact they categorized the research by action categories; including fine motor control, tactile friction, cooperative two-handed tasks, braced two-handed tasks, manipulating a deformable object, tool-assisted tasks and multiple finger tasks. Recent effort has been made to provide visualization taxonomies that can apply to any sense and therefore apply to multimodal applications [Nes05a, CT97]. Nesbitt [Nes05a] developed a framework to support both haptic and auditory displays, but his work focuses on development rather than classification. Whereas Coomans and Timmermans' [CT97] taxonomy was for virtual reality interfaces. Neither of these taxonomies [Nes05a, CT97] address the real importance of data visualization, which is to enable the user to realize and explore the meaning of the information.

This review categorizes the work by the structure of the representation. This taxonomy enables the methods to be uniquely classified. There are seven categories:

- *Charts* cover data that is often displayed visually through charts, line graphs, histograms or scatter plots to display statistical information.
- *Maps* involve data that holds details on geographic or physical spaces, and in visual terms that is represented by maps.
- *Signs* are representations that are instantly recognizable and have associated meanings.

- *Networks* describe the forms that represent associative and relational information presentations. The category includes networks, graphs (from the mathematical meaning), trees and hierarchies.
- *Diagrams* are schematics that illustrate some process, phenomenon, or concept. In visual terms they include circuit diagrams, drawings, illustrations, and sketches.
- *Images & Photo-realistic Renderings* provide realistic representations of real world phenomena.
- *Tables* display information in a tabular form, i.e. in a grid format.

Each of the seven categories will be further explained in the following sections.

This categorization is inspired and extended from two researchers, Bertin and Lohse. Bertin [Ber83] classified visual representations into four categories, namely diagrams, networks, maps and symbols. Lohse et al. [LRBW90] found similar categories (graph and tables, diagrams, icons, maps and network charts) in a study involving the sorting of visual items by subjects from various backgrounds; the results were represented on a 2D graph with one dimension realizing the amount of effort needed to understand the graphic item and the other dimension indicating whether the representation is discrete or continuous.

In this classification and as opposed to Bertin [Ber83], charts are treated separately rather than as being part of the diagrams category. This separation follows the definitions by Lohse et al. [LBWR94], where charts (or graphs) "encode quantitative information" while diagrams describe physical objects or the interrelationships and processes associated with them. According to these definitions, charts and diagrams focus on conveying different information and therefore, the approach for the design of the haptic mapping and interactions will differ. Tables were also treated separately rather than included in charts, as opposed to Lohse et al. [LRBW90], because they display raw quantitative data, which is difficult to convey haptically and therefore will require other forms of designs.

2.2 Charts

Charts are an easy way to visually represent numeric information. In the visual domain, users can easily scan the chart to spot trends, locate maximum and minimum values or drill down to locate specific values. Furthermore, specific values are retrieved through the reference to a legend. However, perceiving charts haptically is a more difficult task: users find it hard to quickly scan over the whole haptic display, and difficult to compare points to a legend to understand their value. This is due to the bandwidth of perception. Many haptic devices present the information through a point-based interface; the user needs to scan the haptic device over the model to comprehend the whole display. Therefore, the challenge is to find effective representations for the numeric data in order to efficiently convey the underlying knowledge. To achieve this aim, different representations and mappings have been developed.

2.2.1 Line Graphs

Line graphs are a common representation form for presenting continuous data and are used in many domains such as mathematics, statistics, and finance.

Representation

Initially, researchers represented the line by a cylinder or an embossed ridge [FB96, YRB00]. These lines were placed on a solid background in a similar configuration to the visual graph. However, users found it difficult to keep the haptic probe on the line and they would slip out of the surface at the edges. Consequently, Fritz and Barner [FB96] applied attraction forces to help the user stay on the line. Yu et al. [YRB00] highlighted that embossed modelling techniques were ineffective for graph exploration and that instead engraved techniques should be used. One year later, Yu et al. [YRBR01] presented an evaluation which showed that the exploration by blind and sighted (but who could only feel the lines) users of graphs made from valleys was more effective (see Figure 6)[1].

Perception of the graph line is only one part of the visualization. Much of the power of line graphs lies in displaying multiple lines on the same graph. Indeed, this enables users to compare values, analyse extremes and calculate intersections. Completing these tasks however requires the user to be able to differentiate the different lines. One method to distinguish these lines is to use surface friction as the discriminating factor by assigning different frictions to different lines; which, was found to be effective [YRB00, YRBR01]. However, with multiple lines there

[1]Readers may also be interested in the work of User Interfaces. In the real world, knobs and buttons have tactile ridges and bumps to guide the user for their use, and provide correct affordances; for example, Miller and Zeleznik [MZ99] discuss various 3D haptic widgets, utilizing notches, dimples and infinitesimal ridges.

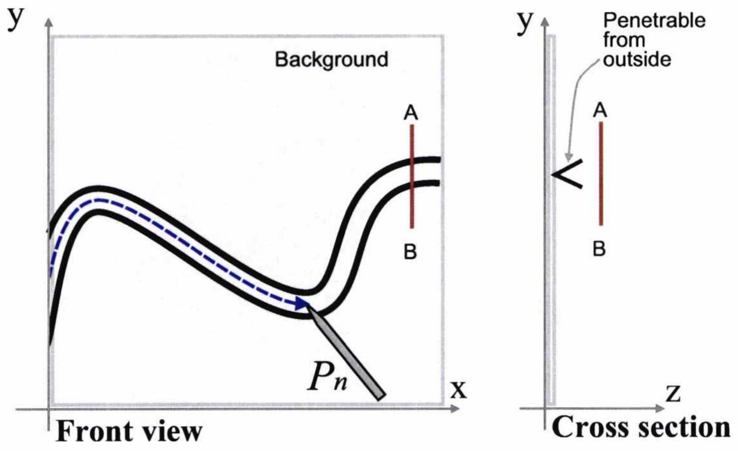


Figure 6: Line modelling technique: engraved line as a V-shape groove attached to the background, after [YRBR01].

can be intersections. Yu et al. [YRB00] took the view that, although friction could help to distinguish the lines it misled users at intersection points.

When the lines become three-dimensional they can be interpreted as 'surfaces'. Many researchers have utilized 3D haptic surfaces, but many of the papers detailing 'haptic surfaces' are about 'haptic rendering' issues, and thus out of the scope of this review.

Quantitative Information

Similarly to visual graphs, a gridline [FB96, YRB00] was used to guide users and provide approximate coordinate values. Although Fritz and Barner [FB96] stated that gridlines, represented as virtual walls which could be felt with a just noticeable force effect, were adequate and necessary, Yu et al. [YRB00] underlined they were rather ineffective and confusing in the non-visual context, saying that they relied on the users' short term memory. This is because the user needs to count the gridlines and remember how many they have passed. They became confused as to which were the grid-lines and which were the graph-lines [YRB00]. Hence, developers have sought other ways to convey values and help navigation. E.g., Fritz and Barner [FB96], and also Yu et al. [YRB00] used a speech synthesizer to aid navigation.

Other haptic and sound research was carried out by Ramloll et al. [RYB$^+$00]. Their work utilized two computers: one to render the haptics using a PHANTOM and the other to produce spatial sounds on headphones. The haptic design (like

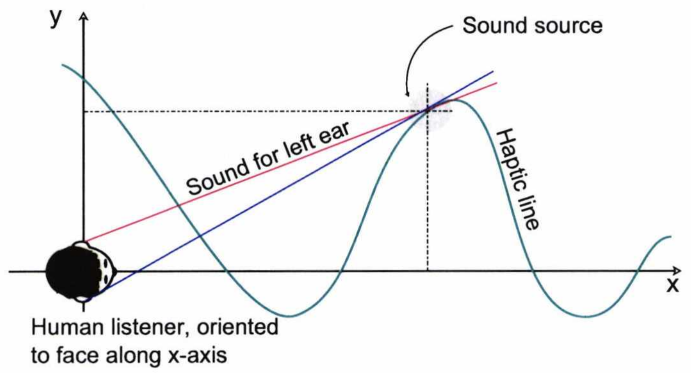


Figure 7: Auditory design, after [RYB$^+$00], where the user is placed virtually at the origin, facing along the x-axis. When stereo sound is used the user is able to hear the difference between positive and negative y-values.

Yu et al. [YRB00]) used grooves to realize the lines on the graph and friction to distinguish the lines. Values were conveyed by speech or by sound at various pitches in the following ways: (1) The Y coordinates of the line graph were mapped to pitch; (2) each curve was accompanied with a different sound, with its intensity decreasing proportionally to the distance from the listener's headphone, with the user's head at the origin (see Figure 7), (3) Speech gave the exact coordinates and an auditory representation provided an overview of the curve by playing pitches successively. This work provides the first steps towards the development of a multimodal system; but included little evaluation.

Yu et al. [YCB02] described a similar tool, targeting visually impaired people, that was evaluated. Their work presents an automatic online graph construction tool that can be opened in any Web browser; it used a pitch-value mapping and the users could feel the line but with the Logitech WingMan Force Feedback mouse^{TM}. The study revealed that the multimodal representation combining audio and haptics is more effective than either modality alone. However, the auditory modality can be limiting; because users do not always have a good *ear* for music and feel isolated when they are using headphones.

Overview and Exploration

The role of an overview is to portray the general trend of the lines as well as depicting relative positions of different key elements in the chart; such as axes, the origin, inflections and any intersections. Ramloll et al. [RYB$^+$00] and Yu et al. [YCB02] used the auditory modality to provide this overview, and Ramloll et al. [RYB$^+$00] guided the user to feel the most important items of the graph such as the display area, axis and curves. One challenge is to keep the user in the display area. Fritz and Barner's 3D surface plots method [FB99] provided one solution. Their surface plot is surrounded with virtual walls (that can be removed) to prevent the user from falling off the surface. Grid planes are integrated as thin penetrable walls producing a small perceivable force. Speech is also used to dictate the user's coordinate position, while friction and texture provide haptic cues of the surface.

Roberts et al. [RFC02] also investigated the haptic overview and proposed different methodologies of exploration: *unguided* exploration, the *constrained* exploration view and the *guided tour*. (1) The unguided exploration enables the user to investigate anywhere in the canvas. Two solutions were tried. In the first one, the user freely explores a bumpmap model of the graph with different depths for each line in the case of multiple curves. The second solution is a 'Friction & Zoom view' where the areas above and below the lines can be felt with different textures. (2) The constrained exploration view is a navigation interaction where a force constrains the user to follow the line. Two solutions are proposed again. The 'Independent graphs' method constrains the user to follow one line at a time where intersection points are indicated by a tactile force. The other lines can be explored by swapping the line under investigation with another one. The 'Exploded Data View' enables the user to view different and simplified views of the information such as a vertical exploration for the maxima and minima, a turning points exploration, the gradient exploration and the intersection points. (3) As for the guided tour, the user movement is constrained. Three different tour metaphors were proposed: a *bus tour* where the user is taken on a predefined route from start to end, a *museum tour* where the user stops at predefined points and then can investigate locally, and a *water skier tour* where the user has some freedom to investigate either-side of the predefined tour.

Automatic Translation

Many researchers have studied the automatic translation of graphics to static tactile representations, see for example, the seminal work by Way and Barner [WB97a, WB97b], while not much research has been done for dynamic haptic visualization. However, Yu et al. [YGB01] described a system that included the full process of haptic graph visualization, from the generation of the graph, its display and finally its dynamic interaction. It was developed in order to increase the accessibility to printed graphs.

The system comprised of a flatbed scanner, a PC and a force feedback device, either a PHANTOM or a WingMan mouse. First the printed graph was scanned and stored as a digital image in the computer. Then, image processing techniques were used to eliminate any irrelevant information and to extract key features for the haptic rendering. The graph was rendered using appropriate techniques, dependent on the type of force-feedback device used. The presented system only worked with simple line-graphs without intersections. Furthermore, Yu et al. [YCB02] dealt with the automatic generation of line-graphs, bar-charts and pie-charts from random data or user-inputted data.

Summary

All previous studies acknowledged that haptic graphs are possible and useful but have their limitations. Many of the challenges arise from the use of single point-of-contact devices, such as the PHANTOM. Some studies [YRB00, RYB$^+$00, YCB02] suggest a multimodal approach, using (say) audio or speech, to overcome the limitations of the device. Moreover, it is acknowledged that the haptic mapping used in these experiments is quite simple [YRB00] or even lacking [RYB$^+$00, YCB02]. Although friction was used to distinguish multiple lines [YRB00], haptics has been limited to represent navigation. Thus, the developers have focused on the end design rather than getting across the underlying principles of the data. After all, the reason for doing a visualization is not to perceive the graphic (graph or plot) but to understand the data itself [Rob00].

2.2.2 Bar and Pie Charts

Bar charts represent a series of categories in a set, each with values. Often the order of the categories is arbitrary, whereas the heights of the bars represent the data-values or the relative values. Visually they are efficient because a user can scan their eyes across the top of the bar chart and pre-attentively view which is the highest and lowest value being presented. Pie charts are included in this section because they represent the same type of data entity as bar charts [Bro92].

Representation

Drawing inspiration from the haptic line-graph research [YRB00, RYB$^+$00], the common haptic design adopted for a bar chart is that the bars are displayed as enclosures (such to use the Logitech WingMan mouse [YB02a]) or as engraved bars [YB02b]. Enclosures are also used for pie-charts [YCB02]. Recently, Wall and Brewster [WB06b] presented 'tac-tiles', a low-tech method to realize pie-charts. The rotary position is given by the position on the graphics-tablet, with the dynamic tactile feedback being realized by the pin-array on a VTPlayer[2] tactile mouse.

Quantitative Information

As opposed to line graphs, different bars or sections of a pie chart are easy to distinguish. Therefore, the main issues are: how to represent the values, how to enable comparison tasks, and how to provide an overview of the information. Yu and Brewster [YB02a, YB02b] used audio alongside haptics to help visually impaired and blind users achieve these tasks. Synthesized speech was used to provide users with exact coordinate values, non-speech audio demonstrated the overview by mapping MIDI notes' pitch to the values, while haptics was utilized for navigation and used to trigger audio events. This is a similar design to earlier work by the same group [RYB$^+$00]. Using this multimodal system design, Yu and Brewster conducted two comparison evaluations that indicated that the multimodal approach provided better results whatever the haptic device being used. In both evaluations, the participants were asked to explore some charts, compare them, answer a workload questionnaire and then answer questions on the trend of the graph: including, which was the lowest bar, the highest bar and the two closest bars in terms of values.

The first study compared the multimodal approach with an exclusive haptic solution [YB02a]. The results revealed that the WingMan mouse was not effective without audio, whereas the PHANTOM version could successfully convey the information alone. However, the multimodal condition had a shorter task completion-time and a lower perceived workload, and thus performed better overall. Hence, audio may be useful to represent the general trend and give information about the minimum and maximum values, while, in the case of value comparison, haptics was necessary. They also suggested that haptics should be used for navigation and audio to represent precise information.

The results of the second study compared the multimodal approach with traditional paper-based tactile diagrams [YB02b]. It showed that the quantity of

[2]VirTouch's VTPlayer^{TM} is a mouse with two tactile 4-by-4 pinboard displays. `www.virtouch2.com`.

correct answers was greater with the multimodal system especially for the last question which utilized value comparison. The multimodal system provided better performance and accuracy than the traditional paper-based tactile diagrams but required more time and effort from the user; this was mainly because the values were realized through speech. Similar conclusions were made to previous work [YRBR01, YB02a], that haptics should be used for navigation and guidance tasks, while speech and non-speech audio should provide detailed value information.

In a comparison of sighted and non-sighted users, Yu and Brewster [YB03] noticed from the position log-file that blind users from the second study mainly used audio for information extraction and haptics for navigation, whereas the sighted users involved in the first study mainly used haptics for both tasks and audio for confirmation. This can be explained by the fact that blind users are more familiar with audio interaction methodologies through technology such as screen readers. This underlines that the role of each modality depends on the target user community.

Wall and Brewster [WB03], when evaluating the discrimination of three haptic properties – friction, stiffness and texture – highlighted that friction gave the best discrimination results. Wall suggested mapping the value of each bar to friction, which could increase the speed and efficiency of the perception.

The same ideas were applied in the case of the pie chart by mapping the pitch to the proportion of the pie division for audio cues and by using a combination of the rectangular and the elliptic enclosure effects for haptic cues [YCB02]. After a preliminary study, a haptic groove was added to indicate the circumference of the pie (see Figure 8) and the sound was made shorter and crisper to remove the echo. The study revealed again that the multimodal representation combining audio and haptics is more effective than either modality alone.

Overview and Exploration

In order to improve exploration, the concept of *beacons* was introduced by Wall and Brewster [WB04]. The concept is equivalent to making notes in the margin, and provides an external memory aid. This is a general technique that could be applied to different types of charts but was evaluated with bar-charts. A traditional qwerty keyboard was used to assign the beacons to bars, with a seek function to haptically drag the user to the beacons using an elastic spring metaphor, and audio cues were used to confirm when a beacon was placed.

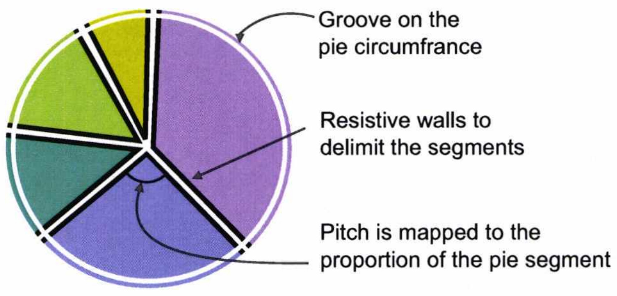

Groove on the pie circumfrance
Resistive walls to delimit the segments
Pitch is mapped to the proportion of the pie segment

Figure 8: The figure depicts how a pie chart is represented haptically, with resistive walls to form an enclosure for each segment, and a groove round the edge to provide an overview of the information, after [YCB02].

Overall, most of the blind participants, especially beginner users, found the beacons potentially useful, particularly for displaying complex data or to quickly move through large data sets. However, the experiment showed it was difficult to simultaneously use the keyboard and the PHANTOM. A proposed solution was to use the numeric pad to find a specific bar by pressing the corresponding number, similar to a CD player remote control which skips tracks, avoiding the placing of beacons [WB04]. However this solution restricts itself to bars.

Another exploration methodology was initially presented by Wall [Wal05] and later extended and evaluated by Wall and Brewster [WB06a], which they named *semi-passive* exploration. Wall [Wal05] proposed a dual interface approach, where the dominant hand navigates with one input-device, such as a mouse or the keyboard, and the other hand or another part of the body receives the feedback. The prototype [WB06a] used a Wacom Intuous 2 graphics tablet augmented with tangible X and Y-axis and the VTPlayer tactile mouse. The user could control the pen with their dominant hand using the tablet, with the VTPlayer mouse being used as a tactile display in the non dominant hand. The prototype was evaluated with bar-charts, where the user could feel the edges of the bars. Interviews with the five visually impaired participants provided positive feedback about the exploration usefulness and highlighted that more verbal guidance was required, in addition to the bar names and their values, as well as solid filled bars instead of the edge representation. Subsequently, the prototype was enhanced by filling the bars and adding contextual speech feedback (dictating whether the user was on, between or above a bar, or outside the graph) [WB06a], but has not yet been re-evaluated.

Automatic Translation

As described in section 2.2.1, Yu et al. [YCB02] developed an automatic online graph construction tool that included line graphs, bar charts and pie charts.

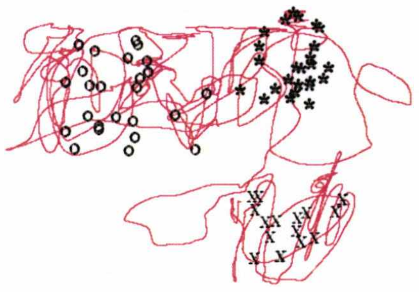
2.2.3 Scatter Plots

Little research has been carried out for the haptic visualization of scatter plots. Haase and Kaczmarek [HK05] designed and tested the display of scatter plots both on a fingertip- and an abdomen-based electrotactile display. Scatter plots were created from sampling bivariate normal distributions with 25 data points and no axes were represented. They conducted an experiment evaluating the electrotactile displays with 12 participants for a correlation estimation task. The performance of the tactile displays was fairly good, although not as good as the visual condition. The abdomen display performed slightly better than the fingertip one but not significantly. They conducted different analysis which suggested that improving several factors on the tactile display would improve the performance for the estimation task, with factors including the sensory quality of the electrotactile stimulation, finding the optimum waveform, and the transfer of information across the electrode-skin interface.

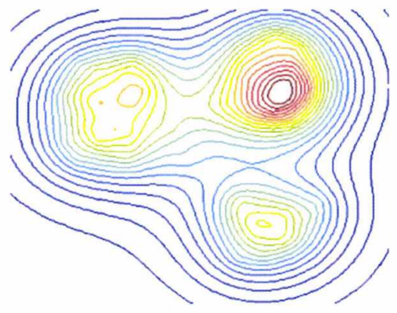
Crossan et al. [CWMS04] on the other hand presented a visualization method based on haptic textures produced using granular synthesis (see Figure 9) and using the force-feedback PHANTOM device so that the user explores the textured surface that represents the plot. Each class of points is displayed using a common audio file associated to each point with the same form of density function. However, the discriminability of signals generated by granular synthesis has not been evaluated.

Summary

One of the challenges when developing haptic versions of charts is that traditionally developers have focused on conveying the shape of the graph, rather than representing the underlying data. Another problem arises through the limitations of the haptic device; that the device only depicts a small area of the whole representation. To overcome these inadequacies, McGookin et al. [MKB05] proposed some guidelines. They suggested providing multiple views of haptic graphs, with each view presenting specific information through a single haptic property and answering a specific user question, while making sure that the different views are

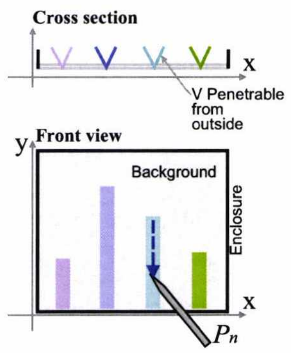


(a) Data Points.

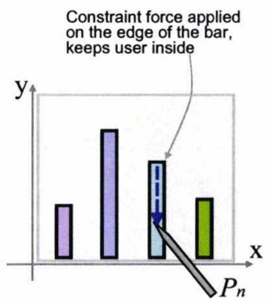


(b) Haptic Scatter Plot

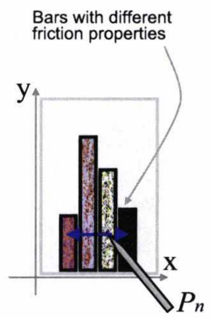
Figure 9: Haptic scatter plot from [CWMS04]: a) data points and cursor trace from a user exploring the granular synthesised scatter plot, b) a contour plot of the Gaussian distributions around the data points.



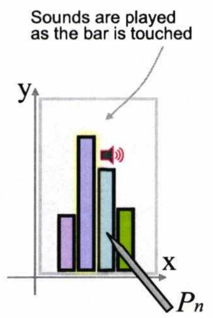
(a) Engraved bars.


Constraint force applied on the edge of the bar, keeps user inside
y
x
P_n

(b) Enclosures.



(c) Friction.



(d) Audio.

Figure 10: Illustrations of different haptic bar chart techniques. 10(a) Engraved bars enable the user to explore the information [YB02a, YB02b]. 10(b) Enclosures, such as created by attractive forces, have been used to keep the user on the bar [YB02a]. 10(c) Friction can be used to distinguish the bars [MKB05]. As the user moves horizontally they gain an overview of the values. 10(d) Auditory information has been used to represent the values of the bars [YB02a, YB02b]

"collocating and overlapping". Lederman and Klatzky's exploratory procedures were envisaged to communicate the different views. For instance, the lateral movement associated with friction could be used to give an overview of the bar chart by attributing different frictions to each bar (see Figure 10). In addition, audio was advised to be integrated into the system to complement haptic. This idea is complimentary with the 'Exploded View' of Roberts et al. [RFC02], who proposed the simplification of different features of the data by separating them into different views.

2.3 Maps

Maps communicate spatial meaning, geographic or physical space. Maps have an explicit and clear association from a point on the map to a specific point in reality. For example, a tourist map of London would show the relative positions of each of the major landmarks. Various researchers have investigated haptic visualizations that could be named maps; they are sub-classified as (1) statistical maps and (2) virtual representation of real environments. Statistical maps are referred to as 'Cartograms' in the latest classification of Lohse et al. [LBWR94], which are maps that show quantitative data.

2.3.1 Statistical Maps

An example of statistical map was developed by Jeong and Gluck [JG02, JG03, Jeo05]. They conducted several studies to evaluate the effectiveness of multimodal choropleth maps. The choropleth map enables geographical regions to be represented by colours, where the colours realize statistical quantities of that region. The haptic display was common to each study; it consisted of different forces of vibration with periodical effect, available from the Immersion^{TM} plug-ins, conveyed by the force-feedback Logitech iFeel^{TM} mouse. Audio was used to represent different quantitative values. The first study [JG02, JG03] looked at bivariate maps and measured completion time and recall rates to compare four models: colour-colour, colour-auditory, colour-haptic and haptic-auditory. The results revealed that the haptic-auditory display provided higher recall rates while, in regard to the completion time, the "haptic displays seem to interfere with other modalities" [JG02]. The second study looked at trivariate maps [Jeo05] and compared colour-only and colour-haptic-auditory displays. Their results showed that completion time was similar for both displays, whereas users could recall the multimodal displays better than the visual counterparts.

2.3.2 Virtual Representation of Real Environments

Various researchers have focused on using virtual maps, primarily to help visually impaired users learn how to navigate in a real environment. Different types of environments have been represented according to the map scale such as indoor environments, local outdoor environment (city) and global outdoor environment (country). Although the environments slightly differ, the holistic approach remains the same, which is to convey the location, size and forms of different objects and the possible paths to reach them.

Representation

There has been much research into the generation and use of (static) tactile maps realized on thermoform or swell paper. These static raised-maps provide an invaluable resource to those with impaired vision; such as work by Gardiner and Perkins [GP96] on representing tactile maps for blind golfers. However tactile map generation is out of the scope of this review because they are static forms, made into a hard copy, rather than dynamic and interactive forms, which are the focus of this chapter.[3]

Schneider and Strothotte [SS00] presented two 2D methods where blind users could create haptic maps guided by the computer. First, using image processing and tracked physical *bricks*, users placed the bricks on a table to make the route, guided by synthetic speech commands. Second, users could follow the routes mapped out on a virtual engraving using the PHANTOM device. The usability of the haptic method was not evaluated. Parente and Bishop [PB03] presented the Blind Audio Tactile System (BATS) using low cost force feedback devices such as the Kensington Orbit3D^{TM} tactile trackball and the Logitech WingMan Rumble^{TM} gamepad. On a map of North Carolina users could feel subtle bumps for county boundaries, large bumps for state boundaries and a constant vibration on cities. An informal observation of four blind high school students interacting with the system highlighted they could successfully use it.

[3]More information about (static) tactile maps can be found in [JMRU06, WB97a, WB97b] and through resources, such as the RNIB National Centre for Tactile Diagrams `www.nctd.org.uk` and the American Printing House For The Blind `www.aph.org`.

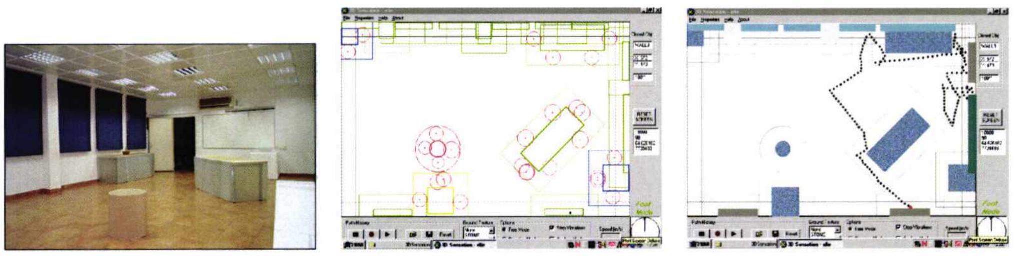
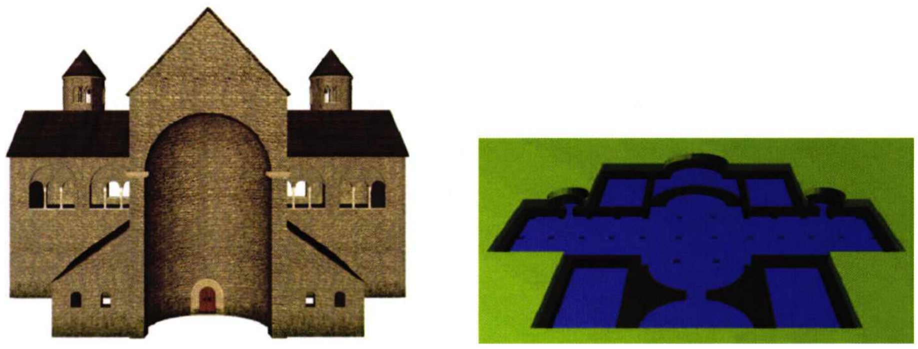

(a) The real environment (b) User interface (c) Teacher interface

Figure 11: The real room and the different interfaces for the *Learning mode* in the haptic VE [LM00]. In b) the user navigates receiving force and audio feedback; i.e. the red circles indicate areas that trigger the guiding agent's intervention for spoken information and in c) the Teacher interface enables the monitoring and recording of the user's path and their actions, which can be subsequently played. Reproduced with permission.

One common approach when designing haptic virtual environments is to use 3D worlds. Lahav and Modiuser [LM00, LM03, LM04] developed a 3D multi-sensory virtual indoor environment using the Microsoft Force Feedback Joystick. The system (see Figure 11) comprised of two modes, the *developer/teacher mode* enabled users to build 3D environments and lay out various haptic and audio effects, while users could navigate and receive feedback in the *learning mode* and be guided around objects by magnetic forces.

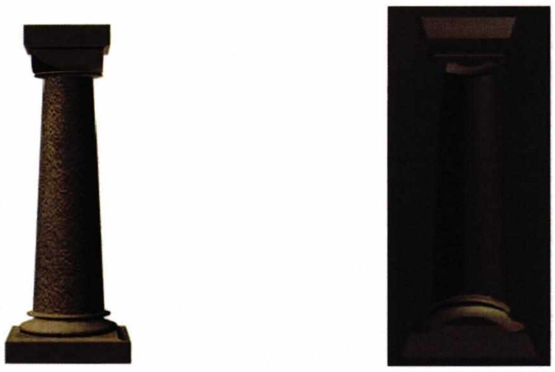
König et al. [KSS00a] adopted a different approach to convey architectural models. They introduced *non-realistic haptic rendering*: techniques that simplify the information to convey its salient features. One idea was to select a number of horizontal cuts through a 3D building, these building-outlines were stacked together to form a $2\frac{1}{2}D$ structure (see Figure 12); this provided an understanding of the building layout but lost height information. Hence, a second model cut the building vertically, such that the user could feel the building in a scalloped form (see Figure 13). Springsguth and Weber [SW03] utilize similar concepts with the structure of the map formed from the ground layer (see Figure 14). Moreover, they assign appropriate waveforms and textures to help users distinguish objects. None of these two representation techniques have been evaluated.

All the aforementioned maps presented static data. However, Magnusson and Rasmuss-Gröhn [MRG04] developed a complex virtual map of a real-world traffic environment presenting not only static objects but also dynamic objects of animated cars and bicycles, see Figure 15. They evaluated their tool on ten visually



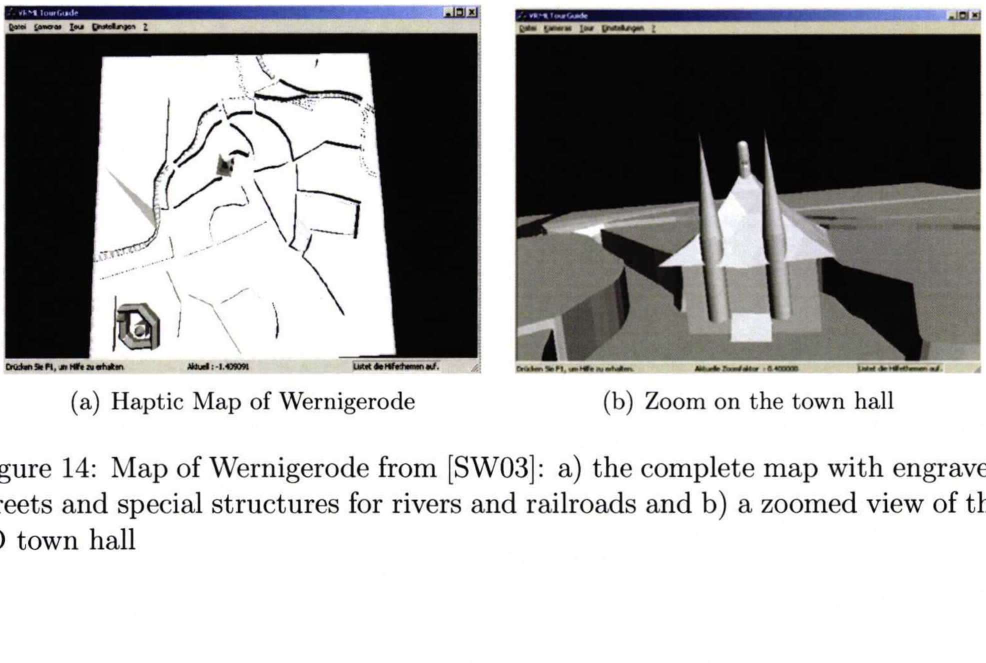
(a) Virtual Castle (b) Ground representation

Figure 12: Foundation representation from [KSS00a]: a) the virtual palace of Otto the Great and b) its simplified $2\frac{1}{2}D$ representation.



(a) Original (b) Engraved

Figure 13: Detail representation from [KSS00a]: a) the original column and b) its engraved representation in which the level of detail can be varied.



(a) Haptic Map of Wernigerode (b) Zoom on the town hall

Figure 14: Map of Wernigerode from [SW03]: a) the complete map with engraved streets and special structures for rivers and railroads and b) a zoomed view of the 3D town hall

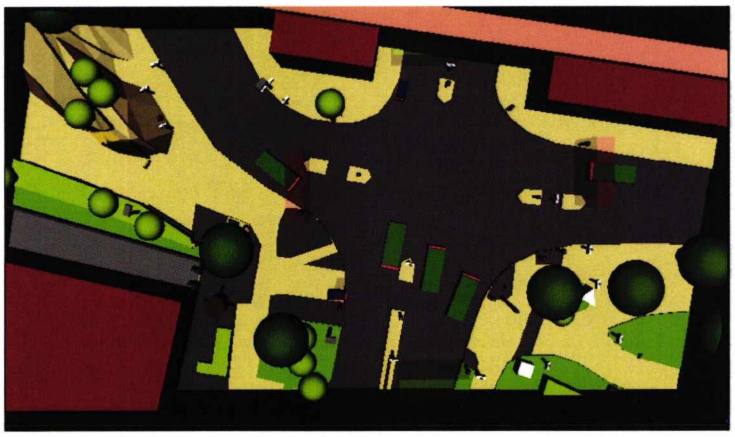


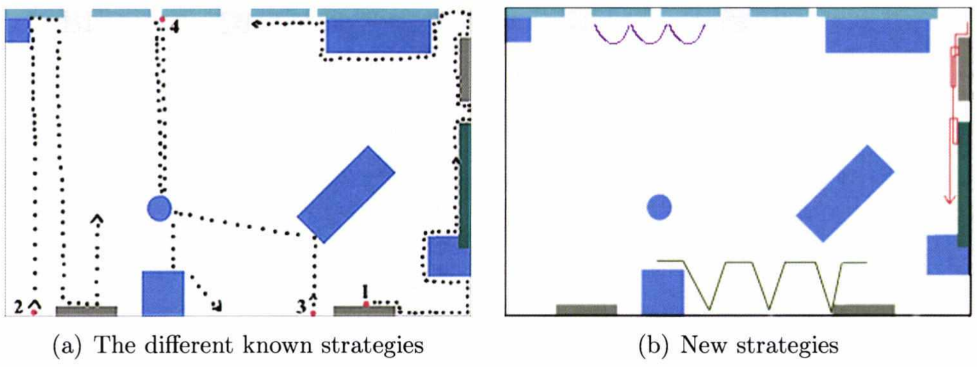
Figure 15: The figure shows the haptic virtual model of Lund in Sweden [MRG04]. Cars are realized by the green boxes while bicycles by blue boxes. Users could navigate using the PHANTOM and hear sounds of the environment; when hit, objects such as cars and bicycles also emitted an appropriate sound. Reproduced with permission.

impaired users and found that the majority of users could navigate both the complex virtual environment and the real world scenario, and that those who could use the cane well in the real world performed well in the virtual world.

Exploration

Map exploration is an important task and has been the focus of various researchers' work. In fact, different researchers have proposed a variety of techniques. Free exploration, where the user actively explores the model without help, is utilized by many developers. For instance, Jansson and Pedersen [JP05] used the VTPlayer mouse to realize a raised version of the USA states. Lahav and Modiuser [LM04], using the haptic environment as shown in Figure 11, performed a study involving 31 blind participants to compare real and virtual exploration processes. They concluded that the haptic virtual environment was effective in enabling users to develop a spatial cognitive map, which could then be transferred to the real space, and additionally it encouraged novel strategies for exploration (see Figure 16).

Magnusson and Gröhn [MRG04] extended the basic free exploration with a technique that enables the user to move around and then be instantly teleported to another place on request. They present a world where the user can freely explore in a virtual box, but when the user pushes the walls of this box, or hit a key on the keyboard, he/she is moved to another virtual world.

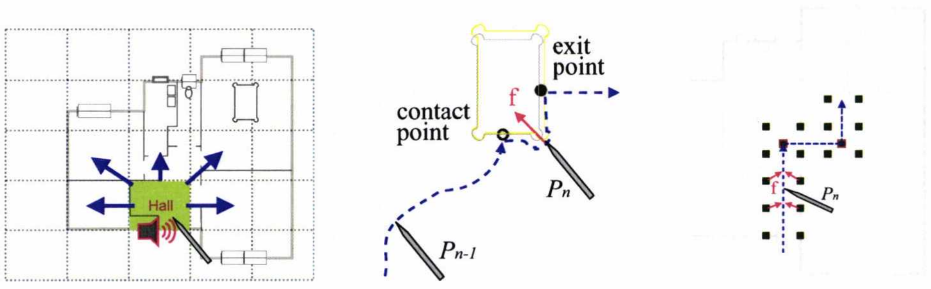


(a) The different known strategies (b) New strategies

Figure 16: a) existing known strategies: (1) the *perimeter* strategy; (2) the *grid* strategy; (3) the *object-to-object* strategy; (4) the *points-of-references* strategy and b) new exploration strategies [LM04].

However, navigating a non-visual environment is difficult, therefore many researchers have designed exploration techniques with some constraints on the movement in addition to free exploration to help the user. Pokluda and Sochor [PS03, PS05] categorized four exploration techniques: *restricted* where the user is guided by the device and fully constrained to a path, *with decision* where the user is offered a choice of possible places of interest to be taken to, *help on demand* which permits the user to freely move around but get additional help when required (for instance when the user feels lost) and *free*. As knowing where to explore next can be difficult; Pokluda and Sochor [PS05] also presented a sphere-menu where the position of other objects could be indicated to the user to help in choosing the next direction to explore. They conducted several experiments with both sighted and blind users and obtained various results about the usability of the different exploration techniques [PS03, PS05]. They concluded that all the techniques should be available to answer individual needs and preferences.

As well as these exploration modes, König et al. [KSS00b] present *active* and *passive*, *orientation* and *navigation* strategies and provide examples with the 3DOF PHANTOM 1.5. The passive methods utilize speech to support the user, while the active methods provide forces to guide and help the user explore. For one passive feedback method, they first divide the scene into a grid then present acoustic information of the contents of the current cell. Users can then decide which cell to move onto, see Figure 17(a). While active methods provide additional forces, such as force fields, to guide the user through predefined paths (see Figure 17(c)) or to enable users to explore specific objects (see Figure 17(b)). These exploration strategies have not been evaluated; however the authors report



(a) Passive orientation. (b) Active navigation. (c) Active navigation

Figure 17: Some examples of passive and active orientation and navigation, after König, Schneider and Strothotte [KSS00b]. 17(a) Passive orientation – the user can freely move room-to-room and be led around the world through audible cues (e.g., a user heading towards a 'kitchen' may hear plate and cutlery sounds, while they may hear an alarm clock when moving towards the bedroom). The volume may increase as they get closer to the room. 17(b) Active navigation – when an object is touched a force-field will help to keep the probe on that object. 17(c) Active navigation – small marks placed through the scene designate the path; attractive forces guide the user.

that the informal tests carried out showed promising results.

Qualitative Information

Researchers have mostly conveyed qualitative information on maps using the auditory modality. For example, speech has been used to speak out the names and descriptions of objects [LM03, SS00, SW03, PB03] and also the directions [LM03, KSS00b, SW03]. Auditory icons have also been used to provide information about the objects or navigational information. Lahav and Modiuser [LM03] associated windows to bird chirping while Parente and Bishop [PB03] attributed a specific sound to each map item, such as traffic sounds for cities and bird chirping sounds for forests, emitted when near the cursor, additionally, they conveyed the direction and distance of cities through the direction and volume of the environmental sounds. Finally, Magnusson and Gröhn [MRG04] played environmental sounds when users bumped into dynamic objects.

2.4 Signs

Signs are something that 'stands for' something else [Cha02]. They may be understood because their meaning is literally denoted by itself such as being a photographic representation of an object, or implied by the environment, such as a 'no right-turn' traffic sign, or merely arbitrarily assigned. Signs (semiotics) have long been studied; more information can be found in the general semiotics literature [Cha02]. More specifically Gumtau [Gum06] discusses haptic interfaces in the context of semiotics. Thus there are many types and forms; three levels of representation are introduced to categorize haptic signs: (1) *monosemic* objects such as icons that have a single meaning, (2) *structured monosemic objects* such as *earcons* and *tactons* that again have one meaning but are structured forms, such that alike objects or actions have a similar appearance, while (3) *polysemic objects* are symbols that have multiple meanings such as *haptic glyphs (hlyphs)*, which are structured signs that realize multiple values, as they include multiple parts, have many ligaments and can provide quantifiable information.

2.4.1 Monosemic Objects - Haptic Icons and User Interfaces

Haptic icons convey a single meaning and have a single association. They, like their visual icon counterpart, represent an idea by convention, association or resemblance. E.g., an icon on a computer interface has one program that is loaded when the icon is activated. MacLean and Enriquez write "...[haptic icons are] brief computer-generated signals displayed through force or tactile feedback to convey information such as event notification, identity, content or state" [ME03]. In fact, their work has resulted in the Hapticon Editor [EM03] and more recently extended into the Haptic icon Prototyper [SMMC06] (see Figure 18). These tools were specially developed to help design haptic icons in terms of waveforms with adjustable duration, frequency and amplitude for each waveform. The haptic icons can be created, by recording the user's 1-DOF knob motion, appending waveforms or by superposing existing icons.

Visell et al. [VLC09] also developed a Haptic Icon Designer application (see Figure19(a)), for the design and playback of vibrotactile icons for information display via floor surfaces, which have not been much investigated. The application allows the design of icons in terms of short-time stimulus using frequency, duration, harmonic content, roughness, and amplitude temporal envelope; and

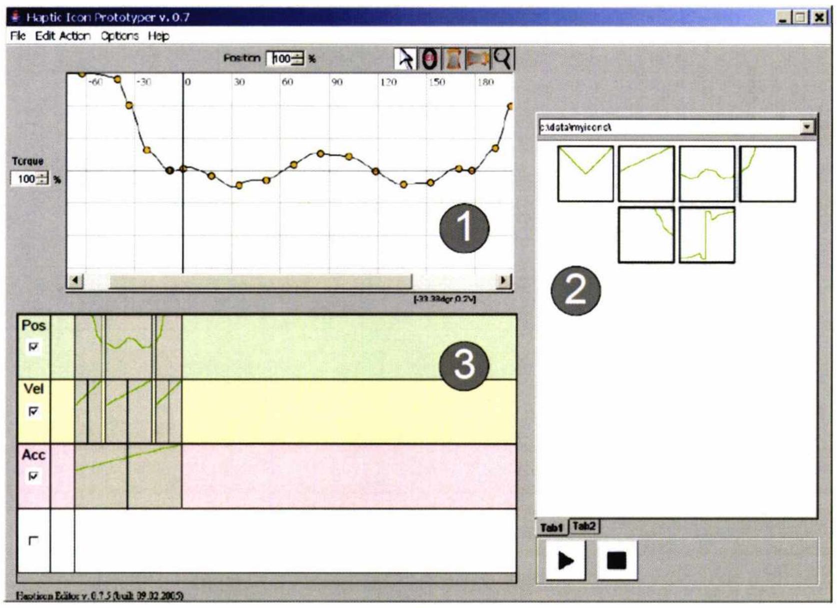
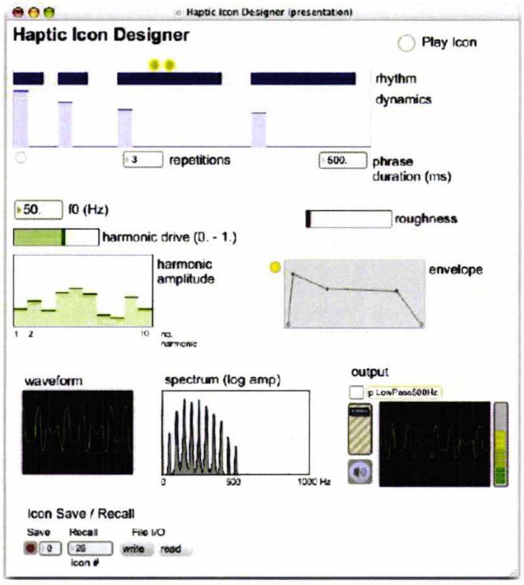
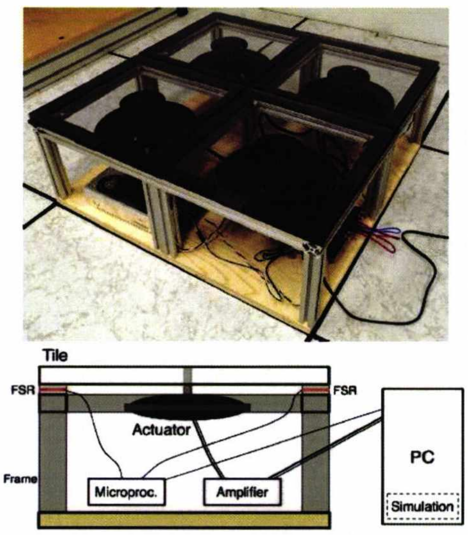


Figure 18: Haptic icon Prototyper [SMMC06]: region (1) is the waveform editor representing the haptic signal and where the waveform can be simply changed by moving control points, region (2) is the tiles palette containing basic haptic effects and region (3) is the tile panel enabling to combine the haptic tiles. Reproduced with permission.



(a) Haptic Icon Designer.



(b) Floor device.

Figure 19: Haptic Icon Designer application (19(a)) for the design of vibrotactile icons displayed on a floor device (19(b)), after [VLC09].

of longer-time structures using a musical phrase metaphor with rhythm, duration, note amplitude and repetition parameters. The application also enables the playback of the icons on their custom floor device (see Figure 19(b)), composed of tiles resting on a rigid substructure, with a vibrotactile actuator attached to the centre of the underside of each of the tiles. A pilot study conducted with eight participants and eight vibrotactile icons developed with the interface, lasting on average 28 minutes, led to a correct identification rate of 55%. However, no guidelines as to which design parameters offer the best recognition could be drawn, but upon suggestions of the participants, future work will examine stimuli in the form of "impact transients" controlled with hardness and contact shape. Lee et al. [LRC09] argued that "Although this low-level access provides great flexibility for shaping the waveform, composing [...] icons in this way is far from being intuitive and is also very time-consuming". Therefore, they present a graphical authoring tool, "VibScoreEditor", that can facilitate the design of vibrotactile patterns, based on a musical metaphor, using a vibrotactile score (to represent the desired pitch, strength, and duration of a vibrotactile note) and a vibrotactile clef (to define how each note is related to physical parameters to generate vibration). Using known piano scores and guitar tablatures, the tool aims at being intuitive and easy to use, even for non-experts [LRC09].

In the medical domain, Ng et al. [NMF+05] investigated the use of haptic icons for situation-awareness in clinical operating rooms by reducing noise levels with a silent vibrotactile alarm system. Two tactors placed on the forearm were used to represent a change in a physiological signal, such as heart rate. The experiment, in a simulated clinical environment, showed that the overall performance was significantly better for the vibrotactile only condition for the identification rate in comparison with an auditory-only and a combined alarm. Users mostly preferred the vibrotactile display as they found it was the best method to attract their attention but the prototype did cause some discomfort.

Much of the work in the literature, in regard to haptic icons, has focused on the evaluation of the effectiveness and discrimination of the icons. Particularly, much work has been done by Karon MacLean and her group at the University of British Columbia. MacLean and Enriquez [ME03] studied the perception of the wave shape (sine, square, etc.), frequency and force amplitude of haptic icons using Multidimensional Scaling (MDS). Their results showed that frequency is the dominant parameter that should be varied between 4 and 20Hz, followed by wave-shape and force magnitude. Chan et al. [CMM04] evaluated the detection and identification in the case of a turn-taking collaborative environment using the Logitech iFeel mouse. Seven haptic icons were divided into three families each having a vibratory pattern (periodic vibration, buzz and taps) using frequency, magnitude, duration and the numbers of signals. They reported that haptic icons could be learned, detected and identified correctly in an acceptable amount of time under different workloads. Finally, Pasquero et al. [PLLM06] evaluated appropriate haptic properties of haptic icons, they used the Tactile Handheld Miniature Bimodal (THMB) interface with eight piezoelectric actuators. The user scenarios included list selection, scrolling, direction signalling (finding a spatial target) and background status notification. Their conclusions indicated that the haptic icons for list selection should be designed using waveform, duration and direction; for direction-signalling direction should be used, while for scrolling the parameters should be speed, direction and waveform. In a study, Enriquez et al. [EMC06] showed that the consistent association of arbitrary meaning to haptic 'phonemes', haptic stimuli, could be learned and recalled up to 45 minutes after training.

In regard to User Interfaces, various researchers have utilized haptic sensations to provide virtual feedback when a button is pressed, or to indicate other processes or commands. These are again monosemic objects as they associate with one piece of information. Van Esch-Bussemakers and Cremers [vEBC04] utilized a complex setup of four PHANTOMs, with their motion restricted to a straight line, along with vibrotactile actuators integrated in the gimbals to simulate a mobile device. The setup was used to experiment how users could access a multidimensional music database. Different types of graphical visualization (wheel, timeline, map and lexicon) could be chosen and force feedback was mainly used for navigation and selection of songs. Some vibratory patterns were implemented to convey messages such as the notification of a system message or that a key/button is unavailable. An evaluation with six users showed that speech was preferred when users searched the database, with the haptic interface for selection and navigation tasks. Lee et al. [LDL$^+$04] presented a different haptic interface. They developed a haptic pen that provided tactile feedback of buttons; it used a solenoid to simulate buzzes and clicks. They mention an illustration where the strength of the buzz could change depending on the closeness to a target object. Informal usage was promising but no evaluation has been conducted. Further to this work, Lee et al. [LPLR07] used a five coin-type vibrating actuator on the fingertip, to add haptic information to a wearable and augmented reality context (AR), so that a visually impaired person could identify specific pictograms or signs. However the system has not been evaluated.

There are a few challenges to overcome when using haptics in non-visual interfaces, especially haptic icons and buttons. First, the user needs to search for the information [Sjö01b], for instance, Sjöström [Sjö01b] proposed a simple cross hair cursor as a solution to enable the user to scan the X and/or Y-axis for the target icons. Second, the user needs to discern that objects are distinct. For instance, Lee and Hannaford [LH03] experimented on "haptic thresholds of the index finger ... for two icon alignments" discovering that the thresholds ranged from 15 to 24 milliNewtons, and that finger motion would affect the threshold.

2.4.2 Monosemic Objects with Structure - Tactons

Drawing inspiration from the visual domain (icons), but in particular, the audio domain (earcons), tactile icons named *tactons* were designed by Brewster and Brown [BB04]. Vibrotactile devices were used to generate the signal, while the information was encoded using frequency, waveform, amplitude, duration, rhythm and spatiotemporal patterns. Three types of tactons have been proposed: *compound tactons*, *hierarchical tactons* and *transformational tactons*. Compound tactons aim to create a language. For example a gradually-increasing high-frequency pulse could represent loading a file, while a similar, but dual frequency vibration could represent saving a file. Hierarchical tactons are tactons that can inherit properties from their parents in a tacton tree. Finally, transformational tactons have several properties each mapped to a different tactile parameter. For instance, a file contains attributes such as size, location, date, which could be mapped onto frequency, body location and rhythm, respectively. Transformational tactons are in fact an example of polysemic objects, and are thus included in the polysemic subsection 2.4.3. Tacton research suggests that rhythm and spatial location were effective parameters for tactons, while the vibrotactile roughness (created using amplitude modulation) was not effective. Consequently, Hoggan and Brewster [HB07] conducted a study to find new efficient techniques to represent tactile texture and they concluded that using different waveforms was the best technique (recognition rate of 94.2%).

Compound Tactons

Compound tactons are often made up of multiple actuators; they can be used to convey geometric information, warning signals, and coded information [vVvE00].

These tactons can be useful in situations where sound, visuals or other senses are already overloaded. E.g., van Veen and van Erp [vVvE00] presented an experiment where they proved that under high G-load conditions, vibrotactile stimulation with a torso display is not degraded whereas the visual channel does degrade or become unusable.

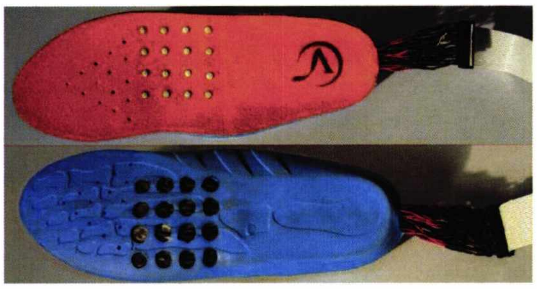
Compound tactons have been applied in various domains. Brown et al. [BBP05] used vibrotactile tactons to encode different types of phone-message call (voice call, text message, or multimedia message). The Engineering Acoustics C2 Tactor^{TM} vibrotactile device was used with modulated sine waves to represent the call types, and the roughness the priority (low, medium or high). They highlighted that rhythm was very successful (with a 93% recognition rate) and that roughness was usable (with 80% recognition rate). Such vibrations have been long used in clocks to tell the time. Töyssy et al. [TRR08] encoded the number ten with long pulses and one with short pulses with a break of two second between the hour and the minute parts (e.g. L-L-S-S-S means 23 and L-S means 11, giving 23:11). After conducting two experiments, they concluded that this mapping was viable and easy to learn. Other researchers have used multiple actuators to encode directions; developing tactile vests, suits, chairs or belts to encode direction.

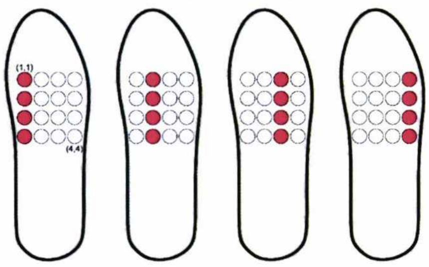
Ertan et al. [ELW$^+$98] designed a wearable haptic vest including a 4-by-4 stimulator array to help blind and visually impaired people navigate the world. Five vibratory instructions (four directions and stop) could be transmitted through the array using patterns. Although there were some problems with the position sensing system, the users gave good feedback. For a similar application, Bosman et al. [BGF$^+$03] presented a wearable wrist-band device that enabled sighted users to navigate real environments through vibratory directional instructions. An experiment conducted with 16 participants highlighted that the system was helpful and "a practical way to deliver guidance information for pedestrians indoors", reducing the number of errors to reach the destination as compared with signage. Similarly, Jansen et al. [JWVG08] designed a tactile shirt display (a horizontal tactor belt and a vertical part on the back), to provide helicopter pilots with quantitative information about altitude and groundspeed when landing in degraded visual conditions also referred to as 'brownout' conditions. They conducted a functional test that highlighted that landing performance in degraded visual environments was significantly improved with the tactile display.

Rupert [Rup00] proposed the Tactile Situation Awareness System (TSAS), a torso suit containing an array of multiple electromechanical tactors and a wearable computer to prevent spatial disorientation in aeronautical or unusual acceleration environments. For instance, the TSAS was used to provide information about pitch and roll for aircraft simulation, or airspeed and velocity in the case of helicopter flights. Overall, users understood the information and achieved good manoeuvring performance; however the limitations of the tactors and the suit did not provide fully effective and comfortable transmission of tactile cues.

Van Erp and van Veen [vEvV01] designed an in-car tactile navigation display with tactile actuators mounted in the seat in order to address visual distraction or workload. A feasibility study evaluated the reaction to messages about a course change where direction was encoded by the location and motion properties and distance by rhythm. Measurements of workload, mental effort and performance under visual only, tactile only and visual+tactile conditions stressed that a vibrotactile display in cars is useful to improve efficiency and safety [vEvV01] as the tactile only condition achieved the best results. Van Erp et al. [vEvVJD05] followed their work by evaluating a vibrotactile belt with eight tactors delivering direction indications to a helicopter and a fast boat pilots. The tactile display was successfully demonstrated in waypoint navigation and even in vibrating environments. In a similar application, Tan et al. [TGYT03] developed a haptic back display which used a 3-by-3 tactor array and also proved that such a display was effective for haptic attentional and directional cueing.

One of the challenges with vibrotactile jackets is that they are tethered to a computer. Recently, Lindeman et al. [LYNH06] have conducted extensive research into wearable vibrotactile displays with the goal of producing a highly reconfigurable, unencumbered, full-body haptic wearable system. They have developed the TactaBox, which is a wireless box that controls 16 tactors through lightweight cables; the TactaVest and the TactaPack (wireless tactor units). Their testing included using the belt from the TactaVest along with the Tactabox, with eight-tactors, in a building-clearing task with 28 participants. An experiment was conducted evaluating the task with and without the vibrotactile cues. The vibrotactile system allowed the user to spend significantly less time clearing a larger percentage of the space [LYNH06]. Jones et al. [JLP06] also developed a wirelessly controlled tactile display, comprising a 4-by-4 array of vibrating motors for the lower back. This system displayed tactile patterns that were tested for navigation in both laboratory and outdoor settings. The experiments indicate that vibrotactile cues or tactons presented to the torso could be accurately and easily identified, reaching 100% recognition rates for most patterns.


(a) Shoe vibrotactile display.



(b) West pattern.

Figure 20: A shoe-integrated tactile display (20(a)) and the display of the 'West' or right pattern (20(b)), after [VBM09].

Velázquez et al. [VBM09] present a shoe-integrated vibrotactile display (see Figure 20(a)) to study how people understand information through their feet. They conducted three studies evaluating the discrimination of direction, shape and pattern with 20 sighted participants. They also studied their application in the case of 3D environment navigation with five participants in a fourth experiment. The results show that participants could understand the direction of motion of dynamic information (left, right, up, down), but could not recognize shapes (square, circles, lines) accurately. For pattern recognition, five tactile patterns were displayed: caution (two intermittent vertical bars), SMS (two times two consecutive vibrations with a pause in the middle), phone call (two long vibrations separated by a pause), relaxation emotion (turning consecutively all actuators off) and exaltation emotion (activating consecutively all actuators). The identification rates averaged to 66% for men and 50% for women, indicating that "people can identify and relate tactile foot patterns to information, familiar signals and emotions". SMS and phone calls patterns were the easiest to identify. Finally, the signals from the direction and pattern studies were combined to deliver navigational cues: 'North' (up) for moving forward, 'South' (down) for moving backward, 'East' (left) for turning left, 'West' (right) for turning right (see the example in Figure 20(b)) and the SMS signal for stopping. Participants achieved good performance overall, with completion times less than 4 minutes, suggesting that "it is feasible to exploit podotactile information for directional navigation in space".

Hierarchical Tactons

An example of a hierarchical tacton is described by Osawa [Osa06] (although he named them tactile glyphs). He utilized the vibrotactile stimulators of the

CyberTouch[4] glove to represent nodes of a hierarchy. Different stimulators on the fingers or palm were activated in turn to encode the position of that node in the hierarchy. Two experiments with 42 and 26 users respectively highlighted that it is better to use the presence of tactile stimulus rather than strength variations as well as a small number of stimuli to improve the accuracy. These experiments intended to find the right parameters to construct discriminable 'tactile glyphs', thus no evaluation within a context of use had been conducted at that stage.

Recently, various researchers have shifted their focus to mobile applications. For example, Hoggan et al. [HAB07, HBJ08] used multiple actuators to provide information about a progress bar and other feedback while entering text on a handheld device. By using spatial location and rhythm for the tactons, their study highlighted that tactile information reduces the error and increases performance as compared to visual interfaces. Many of these systems are 'interfaces' rather than visualizations, because they realize a single value rather than providing any quantifiable information. Thus, additional information on mobile applications can be found in the related areas of wearable computing [Man97, CMH05b, CMH05a, TP01], human computer interaction (HCI) [BMS01] and virtual reality [YJN$^+$96, Sto00].

2.4.3 Polysemic Objects - Haptic Glyphs, Transformational Tactons and Hlyphs

Glyphs in graphical visualization are structures that realize multiple values: they may change their shape, form or size, either in full or in part to realize different values and information. They are certainly similar to tactons. However, tactons (excluding transformational tactons) are used to map an identity (e.g., up, down, left or right), whereas haptic glyphs (or hlyphs) focus on describing quantitative information (e.g., move forward by 30 miles-per-hour).

A good example of a haptic glyph is by Dobson et al. [DBJ$^+$01] who introduce an interface called 'what's shaking'. As the user navigates newsgroups so the amount of postings from individuals is mapped to vibration of a tactile glove and temperature of that glove, such that people perceive "populous groups as warm and active groups as vibrant". An informal test with 12 students showed that they were able to get an understanding of a newsgroup social activity through

[4]CyberTouch^{TM} is the vibrotactile feedback option for Immersion's CyberGlove glove. See `www.immersion.com`

the use of temperature and vibration. Another example is the electronic cane of Hoyle et al. [HFWW04], where vibration increased as the user approached an object. Extensive trials with 27 visually impaired volunteers and other reviewers and advisors led to positive comments about the usefulness of the cane.

Brewster and King [BK05] compared a transformational tacton version of a progression bar with a visual progression bar. They used the Tactaid VBW32^{TM} tactile transducer, and the time remaining was represented by the amount of time separating two vibrations. The vibrotactile progression bar enabled a reduction in time to notice the end of the task of 36% in comparison with the visual progression bar.

Furthermore, Roberts and Franklin [RF05] name the haptic glyphs "hlyphs" and present some design guidelines to enable easy navigation and exploration for (in particular) force-feedback hlyphs. They should be (1) *well structured*, (2) *compound and multifaceted* such that multiple values can be represented, (3) *self contained* such that the user does not confuse multiple glyphs, (4) *endogenous* in design as it is easier to navigate a force-feedback device (such as the PHANTOM) internally, e.g. a valley is easier to navigate than a ridge [YRBR01], (5) enable *pre-attentive perception* such that the user implicitly understands the data, (6) *utilize conceptual mappings* to implicitly understand the mapping function that has been used, and (7) have an appropriate *affordance* such that the user implicitly knows how to operate it.

Recently, Borst and Baiyya [BB07] presented methods for vibrotactile array rendering for enhanced data exploration. In particular, they described the development of a haptic glyph using a 2D tactile array, where tactors form a 5x6 grid. A haptic glyph can be generated by any subset of the following parameters: shape (by specifying list of curve segments), position (2D translation of glyph or 0), orientation (rotation of glyph or 0), scale, count (how many times the shape should be traced), durations (timing for shape tracing) and intensity profile (to modulate tactor intensity). They illustrated the technique by providing examples that include dataset feature extraction in geosciences applications, with a simple mapping to a point glyph's position, and conveying information about the remote user state during collaboration. In the latter case, the position of the remote user was mapped to the haptic glyph position, and the user's orientation vector was represented by the glyph orientation and a line with the glyph shape. Moreover, discrete actions of the remote user, such as joining or leaving a session, or placing a mark on a dataset, were mapped to intensity profiles.

2.5 Networks

Networks describe relational information, including trees, hierarchies and paths, e.g. *a* connects to *b* then *c*. A good example of a hierarchy is blood vessels. Yi and Hayward [YH02] used the 3D force-feedback PenCat/Pro^{TM} interface to enable users to haptically navigate through blood vessels from volume angiograms. The user could see a 2D volumetric projection of the 3D vessel network and zoom in and out of the network. The haptic feedback was used to provide depth information; the harder the user pushes, the larger the forces are. A preliminary study with four students, including the first author, exhibited a mean accuracy of 99.2% and a mean decision-making time equal to 5.4 seconds when discriminating the depth relationship between two separate dots on the cerebral vessel.

Typical network visualizations focus on displaying the relationships among components [LBWR94]. Thus, to understand a network a user must identify each component or node in comparison to its neighbours, and perceive the holistic structure of the chart. Improving the haptic visualization of network charts therefore implies improving these two tasks. In that respect, Jay et al. [JSHG08] developed guidance interactions to improve the "recognition of abstract data at both the micro (individual nodes) and macro (overall structure) level". They used a spring force to guide the user to the leftmost node and a force constraint to keep the user on the surface of the object while he/she explores it. Subsequent nodes could be investigated by pressing the space bar, which moved the spring onto the next node. Audio cues, which were played on demand, indicated the node type: a wind-chime sound identified a sphere while a bicycle bell represented a cube, the two objects used in the experiment. Nine blindfolded (sighted) participants were evaluated under four conditions: namely no cues, audio cues only, haptic cues only and both audio and haptic cues. This was done to evaluate the impact of haptic cues on both the identification of nodes and whether users understood the overall structure. The results showed that the multimodal cues provided the best results, leading to faster recognition of both nodes and structures and more accurate recognition of nodes (100% of correct answers) than in the other conditions. All the participants rated the audio cues as useful for recognizing nodes, and haptic cues as useful for recognizing structure, which validates the design of the role of each cues. This work highlights that constraining the user haptically can help the user understand the structure. Potentially these ideas could be applied to other areas such as Diagrams, Charts and Maps.

Another form of relational and progressive information is music. Although not a dynamic tactile display, Challis and Edwards [CE00] presented a touchpad with tactile overlays to control the output sound. Vacuum formed overlays were used to represent parts of the score, which could be used to control the music. E.g. touching on a bar line would play the music of that bar. An observational study with six users (five sighted and one blind) led to constructive comments to improve the interface (such as avoiding empty space and that a double click was difficult to achieve without haptic feedback). Chu [Chu02], on the other hand, designed the TouchSound software: a digital sound editing software with haptic feedback to enhance interaction and in particular the efficiency of interaction; but did not conduct a user study to evaluate it. Haptic feedback was used to enable users to feel and locate features in the sound such as beats, and users could feel short periodic oscillations or strikes when they were strong. A potential group of users of such an application could be DJs [Chu02] as they need to know the features of the music record to mix it with another song. In fact, Beamish et al. [BMF03, BMF04] developed the D'Groove, a haptic turntable for DJs to enable them to use digital format without losing the feel and advantages of the traditional setup. The DJ could feel the beat structure through virtual bumps at every beat and musical events through friction proportional to amplitude, i.e. the turntable platter or the Q-slider (that replaces the needle) is harder to move in frequency rich moments and inversely. Additionally, the DJ could create new effects with springs and texture. The observational study with six experienced DJs [BMF04] resulted in an overall positive feedback and highlighted the need for a self-contained turntable unit with higher turntable torque and better appearance.

2.6 Diagrams

Diagrams illustrate some process, phenomenon, or concept. In visual terms they include schematic diagrams and illustrations. For instance, a diagram may be used to illustrate the process of constructing a flat-pack cupboard, or could be used in a botanical encyclopedia to illustrate and label the leaf types of different trees. Lohse et al. [LBWR94] distinguish two types of diagrams: structure diagrams and process diagrams. Structure diagrams are "a static description of the physical object. The spatial data expresses the true coordinate dimensions of the object" while the process diagrams "describe the interrelationships and processes associated with physical objects. The spatial data expresses dynamic, continuous, or temporal relationships among the objects in process diagrams". No work, that could be found, has been done on structure diagrams. However, the work done in scientific visualization can be classified as a combination of a structure and process diagrams, as haptics is often used to provide more insight into the physical structure of the objects along with the processes associated with these objects.

2.6.1 Scientific Visualization Diagrams

Taylor II et al. [TIRC$^+$93, TICO$^+$97] used force feedback with a Nanomanipulator for the manipulation of surfaces at nanoscale. This atomic scale teleoperation enabled the users to learn about the surface properties, to more effectively explore it and to interactively modify the surface with near real-time observations of dynamic processes [TIRC$^+$93, TICO$^+$97]. Through collaboration with scientists, force feedback control was found to be useful as it enabled finer control and touch; helping users locate the correct point and the right modification path [TICO$^+$97, TI00]. Various researchers have created molecule docking visualizations. Brooks and colleagues, in the GROPE project [BJOYBK90, TI00], presented a molecule docking example where users could view molecules, interact and feel the different forces from individual molecules. The experiment, conducted with 12 experienced biochemists, showed that 6D rigid body docking manoeuvres were performed faster and that the chemists proceeded more directly toward the correct minimum when using haptics as compared to a visual-only display [BJOYBK90]. Most importantly, the chemists reported getting better understanding of the situation and the problem. Sankaranarayanan et al. [SWS$^+$03] developed a similar haptic display, although not evaluated, to accelerate the learning process in molecular biology. Using the PHANTOM, users could feel the forces of the molecules and view them in augmented reality. Similarly, Durbeck et al. [DMW$^+$98] used a visual and haptic display to represent vector fields. The display could represent various fields including gravity, pressure, force, current, velocity and scalar gradients, and that could be felt with a PHANTOM. No evaluation has been reported. Finally, Wies et al. [WGO$^+$00] used speech and force feedback with the Logitech WingMan Force Feedback mouse to let students find the charge on an uniformly charged sphere in the education of physics, in particular the study of electric fields by blind and visually impaired students. A feasibility study conducted with four blind students and educational experts led to positive feedback about the usefulness of haptics and the system for accessible education.

Haptic volumetric field visualization has been researched by a few research groups. First, Avila and Sobierajski [AS96] used the PHANTOM to convey volumetric information. They detailed both an isosurface and volume rendering of the volumetric data. Their isosurface was calculated as a solid surface, whereas the volume rendering mimicked the gel-like representation of its graphical counterpart, with more opaque material being realized by a greater motion restraint. The system was not evaluated but the authors found "that the integration of haptic interaction into a scientific visualization process can lead to a better understanding of complex, three-dimensional data, and can result in more natural, intuitive methods for modifying this data". Second, Lawrence and Pao [PL98, LPLN04] and their colleagues described various haptic flow-visualization applications. They presented various rendering modes for scientific visualization including those for vector and tensor fields, "which are types of data that are difficult to graphically visualize" [PL98]. For vector fields these included *orientation constraint* that constrains the users' stylus to the direction of the vector, *transverse damping* where forces are applied "in directions transverse to the field .. and forces proportional to the field magnitude", and *relative drag* where forces are applied proportional to the "difference between vector fields values and the user's hand". They have presented various applications of these ideas including a shock and vortex visualization [LLPN00] and work on combined visual/haptic rendering modes [IBL+99], which detailed various techniques including a flow field visualization where the orientation of the haptic actuator was constrained to depict the orientation of the flow, and a constraint method to enable the user to follow the isolines of the flow in a vortex visualization. Although the authors report the benefits of augmenting graphical visualizations with haptics to convey the information, no formal user evaluation testing the usefulness of the rendering modes has been conducted. Reimersdahl et al. [vRBKB03] extended Pao and Lawrence's work [PL98] and presented various haptic rendering techniques for the interactive exploration of Computational Fluid Dynamics (CFD) datasets in virtual environments. After an initial evaluation with four participants, they concluded that adding haptics was "a very promising approach to facilitate the exploration process of CFD simulation data". Mendez et al. [MYN+05] developed two haptic guidance navigation methods to help users locate regions of interest in volumetric data. They provide a 2D solution for the exploration of 3D datasets, by using the Proactive Desk haptic feedback device, which solves the common problem of occlusion in 3D representations, but they have not evaluated it. Lundin et al. [LCP^{+}07] have developed 'haptic primitives' (directed force, point, line, plane) and 'haptic modes', which are high-level haptic interactions definitions (such as viscosity, gradient force, vortex tube) based on the haptic primitives. They have integrated these modes into the Volume Haptics Toolkit (VHTK). A pilot qualitative study was conducted with seven experienced radiologists with the task of exploring the blood flow in the heart and identifying paths using stream ribbons and some haptic modes. It confirmed that adding force feedback was useful for information and guidance and contributed to the understanding of the distributed flow; they concluded that "haptic feedback was considered helpful and the combination of haptics/graphics produced a better result than using visual feedback alone" [LCP^{+}07]. Recently, Palmerius and Forsell [PF09] also tested three different haptic metaphors (line, surface and force) representing volumetric vector data to identify faint structures. The experiment conducted with 12 participants showed that the line and the surface metaphors permit the user to better identify faint structures in comparison to the force metaphor.

2.6.2 Other Diagrams

Various other diagram styles exist. For instance, a block diagram is a simple abstract diagram of a process or system. Kahol et al. [KTMP05] treated block diagrams as images and hence used computer vision techniques to parse and then convey the information through audio and haptics. Speech was used to narrate the features and tactile feedback, via a custom made glove, indicated the presence of block diagrams components and their position: long pulses indicated text, with (x,y) coordinates being mapped to the duration of a tactile pulse. They report that an initial user study led to a 100% recognition accuracy for the shapes of the blocks and that users commented on the tactile feedback as being very helpful. Likewise, Horstmann et al. [HHK^{+}04] developed a system to automatically translate technical diagrams into tactile versions. The user could navigate the information through the keyboard and force-feedback joystick. However, the haptic mapping used is not very detailed and thus it is unclear whether the haptic feedback contributed to the reported users' ability to build up a coherent representation of the diagrams during the evaluation. This work ([KTMP05, HHK^{+}04]) was classified under *Diagrams* but specific use cases could also fit in *Networks*.

For instance, Horstmann et al. [HHK$^+$04] applied their system in three domains: "analogue and digital electronic circuits, certain UML [...] diagrams and architectural floor plans". The architectural floor plans could be classified in the Networks section while the circuits and UML diagrams would classify more as process diagrams as they tend to focus not only on the relationships of elements but also the processes involved. On the other hand, block diagrams can focus either on the interconnections and relationships of elements, thus qualifying in the *Networks* section, or on processes, therefore fitting better in the *Diagram* section.

Finally, researchers have developed various ways to convert semantic web, or other structured diagrams into non-visual forms. Although these are not necessarily examples of dynamic visualization, they are relevant to this section. First, Lorenz and Horstmann [LH04] present a short paper on the 'semantic access to graphical resources for blind users'. Fredj and Duce [FD06] present GraSSML which utilizes Scalable Vector Graphics (SVG) to develop a high-level description language that can be applied to different modalities. Lastly, Krufka and Barner [KB05b] present some technical advice on how to automatically convert SVG into tactile graphics, which were printed to paper using a Braille printer.

2.7 Images & Photo-realistic Renderings

Images and Photo-realistic Renderings provide realistic representations of real world objects or scenes. Way and Barner [WB97a, WB97b] presented two seminal papers on the automatic translation of images into a tactile form, felt with the microcapsule paper. Furthermore, Kurze presented a haptic renderer that converts a 3D scene into a 2D form that can be printed as a raised line drawing [Kur97] or used in some systems for interactive exploration [Kur99]. Although these are foremostly static visualizations, they are important and relevant to this chapter because they utilize aggregation and simplification methods that are necessary for haptic visualization. Additionally, Kurze [Kur99, Kur97] developed a navigation aid (TGuide) to interactively explore tactile images. The system utilized a custom made mouse with eight vibrators, a digitizer pen and a speech recognition and synthesis system. Their custom device could be used in a two-handed mode with one hand resting on the device, or in a one-hand mode with the custom device and pen together. He proposed two guidance modes: a *tourist metaphor* where the user is guided from one object to the next and a *scout metaphor* where the user can ask for the location of an object through simple spoken commands. The guidance was provided through directional hints by activating the corresponding vibrator. An evaluation was conducted with 12 sighted and blind people for a guidance to a target task and compared a visual, an acoustic and a tactile condition. It highlighted that the tactile directional guidance was more efficient than acoustic guidance, and comparable to visual guidance and was judged of "great value" by the blind users.

Various other researchers have developed haptic renderings to realize three-dimensional buildings and images. For example, Pokluda and Sochor [PS05] used the PHANTOM to enable users to explore or be led around a building, and Laycock et al. [LLD06] permit users to navigate a high quality rendering of an ancient hall using a PHANTOM Omni, not only to feel touchable objects but also to change the speed of the tour or the orientation. Whether the generation of high-quality renderings is useful is a matter for discussion, for instance, König et al. [KSS00a] discuss that users find it difficult to perceive shapes, sizes and complex objects effectively and that abstract (non-realistic) renderings are more beneficial.

Other than shapes and sizes of objects, a visual component of images that is difficult to convey haptically is colour. Kahol et al. [KFBP06] investigated colour perception for blind (congenitally and late-blind) people by associating colours to haptic textures. They developed a model where colours are represented by three planes with RGB information mapped to three levels of friction (none, medium, high) on each and an additional 'mixer' plane to tell whether the colour is pure or white/black colour has been added, using the PHANTOM device (see Figure 21). On the mixer plane, the colour added is conveyed by controlling the direction in which the probe can be moved: allowing movement on the left only for black, on the right only for white, and on both sides for pure colour (see Figure 21). Therefore, colours are perceived as a combination of seven basic colours (red, green, blue, yellow, orange, violet, black). As an example, pink is represented by red (high friction on first plane, no friction on the next two) and added white (movement on the right only). They conducted two experiments to evaluate colour perception and to assess the perceived similarity between colours using multidimensional scaling. For the first experiment, all groups (five blind, five sighted but blindfolded, five sighted not blindfolded, and tested again blindfolded) achieved a 100% recognition rate. In the second experiment, the similarity matrix obtained with similarity scores for pairs of colours is scaled to a 2D and 3D space, which is then compared to the HSV colour space, to evaluate how well

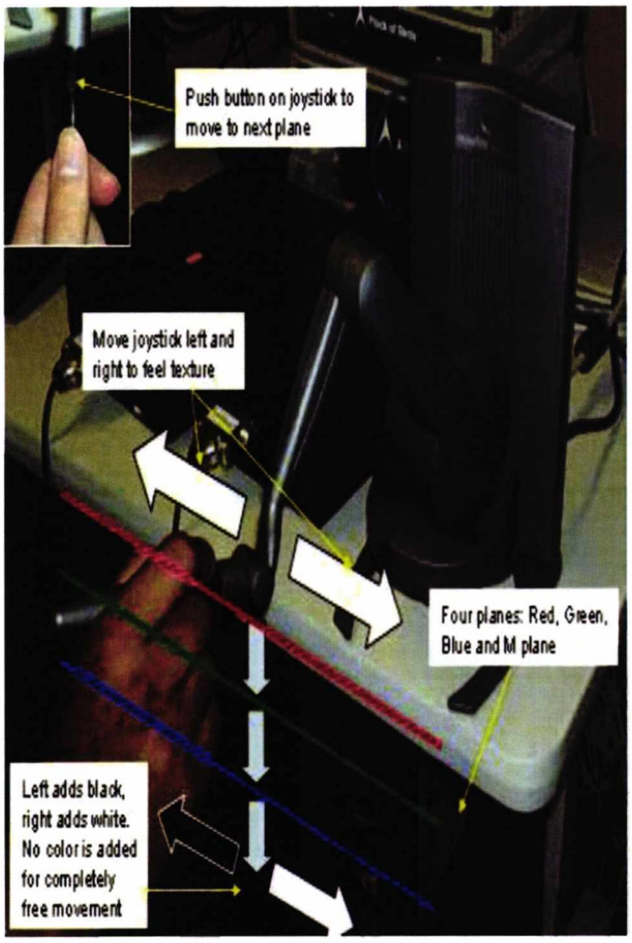
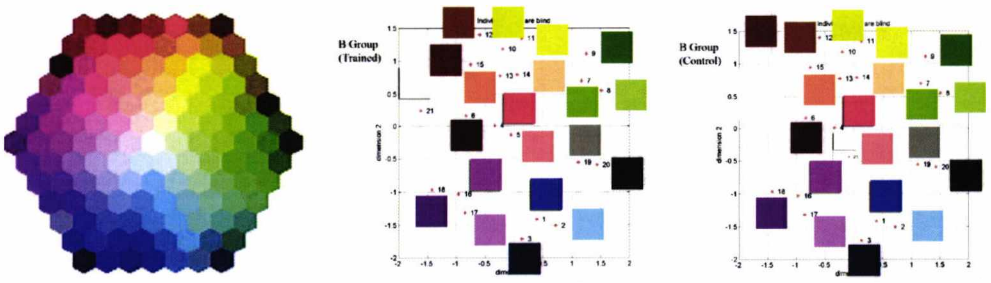

Push button on joystick to move to next plane
Move joystick left and right to feel texture
Four planes: Red, Green, Blue and M plane
Left adds black, right adds white. No color is added for completely free movement

Figure 21: This screenshot, from [KFBP06], represents the mapping between the device movement to the different colour planes.



(a) Colour wheel. (b) Trained group. (c) Control group.

Figure 22: Results for the mapped colour space for the trained (22(b)) and control group (22(c)) as compared to the standard colour wheel (22(a)), after [KFBP06].

the system performs. Two groups of five blind and 15 sighted, trained (completed first experiment) and untrained participants (comparing haptic textures similarities), obtained high congruencies between the 2D and 3D similarity space and colour wheel (see Figure 22). Moreover, no significant difference was observed between the two groups. After obtaining good results validating the model, they developed real-time perception using a CyberTouch glove, with index, middle and ring finger associated to the first three planes, and little finger and thumb for the mixer plane. They used tactons with a sequence of three short or long pulses to indicate the quantization of the colour (i.e. red is long-long-long, which is high quantization, for index finger, no vibrations for the other fingers; and pink is the same as red plus long-long-long on the thumb). The colour is conveyed on mouse movements during exploration. They evaluated the real-time mapping, with the five blind participants from the first experiment, for the exploration of natural images segmented through k-means into 21 colour bins. Five images were used for training and "All the users achieved recognition accuracy of 100% from the first trial itself" [KFBP06]. 35 images were used for testing and the overall accuracy of colour recognition, calculated as the number of times the participant reported the correct colour during exploration, reached 98.5%. In order to evaluate whether the initial mapping needed to be learnt to obtain better accuracy, this experiment was repeated with 15 sighted users who used the direct mapping from colour to vibrotactile cue only. The average learning accuracy was only 78.5% after four trials, leading Kahol et al. to conclude that "This initial result tends to suggest that training that allows users to associate texture (a known concept) with a colour allows for better perception of colour rather than training that requires directly mapping cueing (a learnt concept) to color" [KFBP06].

2.8 Tables

Tables display information in a tabular form, i.e. the data is structured in a grid format. Different kind of information can be depicted in a grid format, including numerical (such as spreadsheet layout) or graphical (such as the reorderable matrix [Ber83, SM05]), or sonified [KB05a]. In information visualization, tabular representations are extremely popular. They display multidimensional data and enable the user to compare values and perform correlations. Various researchers have looked at sonified tables (e.g. [KB05a]), and Kildal and Brewster [KB07] used tactile feedback to enhance their sonified table (TableViz), but to date no haptic visualizations could be classified as representing tabular data. On the one hand, current haptic devices do not lend themselves to tabular representations. This may be because many of these devices are point-based devices, which make it difficult to convey exact values haptically. It is likely that specific modalities are better at displaying certain data; and are most relevant to a particular task. For instance, van Esch-Bussemakers and Cremers [vEBC04] investigated the use of different modalities to operate different functions; concluding "that participants did have preferences for certain modalities when handling certain types of information".

2.9 Summary

This chapter described the state of the art in Haptic Data Visualization (HDV). The scope of the work presented here has been towards data presentation using haptic devices. Two areas that would expand the review substantially are to include the haptic papers from Virtual Reality and Medical Applications. But these subjects are deemed to be outside of the scope of this review because they tend to focus on *user interface* methodologies and represent 'state' rather than 'value'.

The classification of the various papers is based on deep reflection and research; however some of the papers could be placed under several categories. E.g., Yi and Hayward [YH02] present a method to haptically navigate through blood vessels, and thus it was categorized as a *network* technique, but it could be classified as being a *diagram* representation; their example demonstrates a hierarchical form which is better described in the *networks* category. Another challenge is to separate *networks* from *diagrams*. In particular, this affects the work on haptic music. Music could be listed as a *diagram* as it demonstrates a process. However, music fits better in *networks* (see section 2.5) because it demonstrates many structured relations which could be hierarchical. This concurs with the classification by Lohse et al. [LBWR94] who included music under its network category.

By the very nature of this review presentation, it can easily be seen where researchers have generally focused their effort: that most research has been in the area of *charts*, in particular many researchers have focused on line graphs, bar charts and pie charts. Three further areas that researchers have investigated are *maps*, *diagrams* and *signs*. Researchers have looked at virtual representations of real environments, scientific 'diagrams' (such as molecular docking and field flow), and *signs* particularly monosemic objects of haptic icons, user interfaces and tactons. However, significant lack of development has been in the areas of *networks*, *images and photo-realistic renderings* and *tables*.

Finally, the area of haptic visualization will continue to grow and develop, helped along by research in perception especially perception and interference of multimodal solutions, aided by new devices and technologies and inspired by ever complex information visualization solutions.

2.10 Research Challenges

This chapter reviewed the state of the art in HDV, using a classification that aims to highlight representations that are well and under investigated as well as the common methods across developed techniques and their results. Even though such a categorized review is useful for analysis of existing work and an important first step when developing new haptic visualizations, the classification of the papers is perhaps a minor challenge; a greater challenge is *how* appropriate and effective haptic visualizations can be generated. In fact, multiple challenges exist. Developers need to choose appropriate mappings (based on how the user will perceive that information), develop effective models, provide efficient interaction techniques, and utilize the most appropriate hardware. These challenges are discussed below and divided into three issues the developer faces during design (representation, interactions in particular navigation, and hardware) and two holistic solutions (see Figure 23).

Effective haptic data visualizations should be using haptic properties both at the representation level (e.g., engraved lines, bumps) and at the task or interaction level (navigation, selection, manipulation). However, it would seem that many researchers have focused on designing haptic representations that are similar to their visual counterparts rather than finding the best haptic visualization for that task. Although it may be useful to learn from the visual domain, it is better to develop specific visualizations [Rob04] and developers should think carefully how they can best display their information through haptics. Researchers have started to think about the effective design issue [MKB05, RP07, Fra07], but more work is required.

Palmerius [PF09] writes on haptic interaction design saying that it is useful to "not only design the feedback to most effectively convey the information, but to

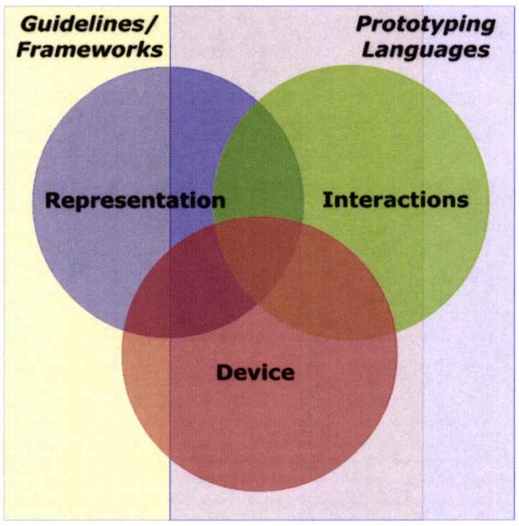


Figure 23: Model Challenges and Solutions.

convey it in an intuitive manner so not to confuse the user". Therefore, interactions, and in particular navigation and exploration tasks, are another important and timely challenge. In fact, it is often difficult for users to understand where they are located or to build a mental model. With visual information, users gain an overview by scanning their eyes over the whole display. Haptically scanning over the whole display is difficult and often impractical. Researchers have utilized additional modalities to give the user an understanding of context, place and value, often relying on speech and sonification (e.g. [RYB$^+$00]). However, providing a haptic alternative can be useful in cases where other modalities are overloaded or unavailable. Some researchers have exploited haptic properties to propose different modes of exploration, where forces are used to help or constrain the movement [RF05, PS03, KSS00b] and highlight important features, but more metaphors and interaction techniques need to be proposed.

Choosing the right hardware (haptic device) is also important and can affect the user's perception of the information. The specific haptic device used by the designer is mentioned for each technique in the review, because in addition to the importance of the type of technology used (tactile or force feedback), each device adds its own limitations and challenges (i.e. mouse vs stylus interactions, 2D/3D, point-based or multi-points). The wide use of vibrotactile tactors and point-based force-feedback devices demonstrates the usefulness of these devices, however, they may not be suitable for the task, and thus researchers should be encouraged to find the right haptic device for the task, rather than merely reaching for the most available one. Researchers should also try to compensate for the specific limitations of the device through the use of new interaction metaphors (passive/active exploration). The next chapter proposes an implementation of guidance metaphors as a solution to improve navigation for the haptic exploration of line charts and to reduce the difficulty of point-based device interaction.

These three issues, representation, interaction and hardware, are distinct but influence each other (see Figure 23). For instance, the type of device (tactile or fore feedback) and its limitations affect the possible design choices for representation and interactions. E.g. a tactile representation is not designed the same way as a force feedback model and even within the same family of devices, i.e. the PHANTOM Omni or Desktop, the difference in force resolution will instigate different techniques. The designed representation will also guide the design of interaction metaphors: different representations have different goals and different information to convey. Furthermore, the interactions required for an application, gesture versus desktop interactions for example, will drive the choice for the appropriate hardware (i.e. gloves versus stylus device). The developer needs to carefully consider design choices for these three issues; nevertheless, how to make the appropriate decisions is not obvious. Generally, the haptic representation and interaction techniques for the tasks are determined on a case-by-case basis (or at least within a category such as Charts, Maps, etc.) and for the available hardware; however ideas for representations and tasks (valleys instead of ridges and exploration techniques) can be reused across applications and research areas and should be therefore investigated.

One approach to promote the reuse of ideas and help in the design of haptic applications is to compile the results of past work into guidelines, rules or even development methodologies. Currently there are few haptic visualization guidelines to help create effective haptic visualizations and they are scattered among the publications. Thus, there is a need to produce and gather effective guidelines for haptic data visualization. However, some guidelines do exist in the more general context of designing haptic interactions, thus including guidelines for haptic data visualization, and they are currently being collated under the ISO Standard for tactile and haptic interactions [vEAC06].

Following on from guidelines, evaluation is an important aspect of developing haptic presentations. Some evaluation studies have been described in this chapter, but certainly more are required. Evaluation enables the testing of whether a design satisfies its goals, such as efficiency or effectiveness in conveying information, and the collection of recommendations to improve it. Such a process – developing a design, evaluating it, improving it and repeating the sequence – to achieve the best design is commonly referred to as prototyping. Therefore, the second approach that facilitates the development of haptic presentations is the creation of prototyping tools for rapid development and testing of design ideas. Similar to the three design issues, these two solutions (guidelines/methodologies and prototyping languages/tools) overlap (see Figure 23). Indeed, prototyping languages/tools allow rapid development, but the development should be done following existing guidelines, rules or methodologies gathered from existing knowledge. On the other hand, rapid development can lead to rapid testing, which in turn can help devising new guidelines or improving existing rules and methodologies. Each solution can help the other and benefit from its results. These two holistic solutions to facilitate the design of haptic applications will be expanded in the second part of the thesis. But first, the design of a line chart application is explored, describing its representation, interaction metaphors and pilot testing, along with the challenges encountered.

Chapter 3

A Line Chart Application

Line charts are one of the most common representations for statistical data. However, many challenges remain in relation to non-visual techniques, such as getting an overview, locating and comparing specific features and dealing with multiple lines, as described in Chapter 2. Researchers resort to solving these using the auditory modality, especially in the case of methods for gaining an overview and exact values. However, sonification may not be appropriate or available, and so a pure haptic technique would be useful in this case. This chapter describes a prototype haptic system, that uses guidance metaphors to help users obtain an overview of information depicted through the haptic modality. These metaphors, with varying degrees of movement constraint, provide different type of information, such as the general line shape through a continuous movement or the points of interest by slowing down or stopping at their locations, which combined help building a mental model of the representation.

This chapter will first introduce the design goals that motivated the development of the prototype line chart application, followed by the details of the implementation, including the development of these exploration metaphors. This chapter will also present the pilot study, that was conducted to evaluate these exploration metaphors in getting an overview, and its results, before concluding.

3.1 Design Goals

One of the remaining challenges in the non-visual exploration of line charts is obtaining an overview haptically. With sight, users gain an overview 'at a glance' and can direct their attention to interesting features while still seeing the whole picture with peripheral vision. However, gaining an overview haptically is more problematic and less automatic. Users need to actively move the haptic device, or be led to points, to obtain an understanding of the information presented. In particular, most commercial devices are single point-based devices, such as the PHANTOM. With such devices, the information transmitted is greatly reduced. Exploring the whole surface requires the users to 'poke' the entire area to build up a mental picture of what they are touching. Thus, this process relies heavily on short-term memory and is therefore quite cumbersome.

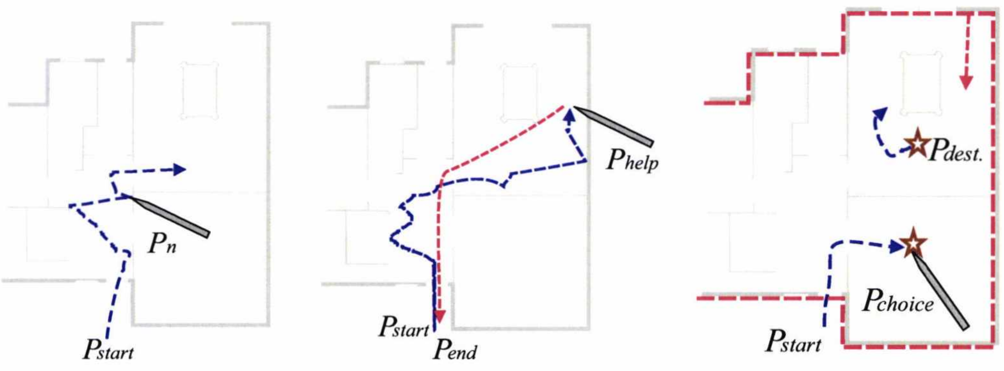
Previous work (see Chapter 2) resorted to the auditory modality to solve the non-visual overview challenge. The sonification is usually obtained by mapping the pitch of MIDI notes to the y-axis, which are played successively when the user is moving the device along the line [RYB$^+$00]. However, previous work in charts presentation lacks methods for providing the overview haptically. Overview is the first step in understanding data, according to Ben Scheidernman's visualization mantra "Overview first, zoom & filter, then details on demand" (Chapter 14, page 539 [SP09]), and as the auditory modality can be overloaded or unavailable, it is important to provide a haptic solution for the overview.

Guidance metaphors coupled with free exploration can contribute to building a better mental image of the chart, thus help to provide an overview, and were therefore explored. Indeed there are two common forms of haptic exploration: free exploration and guidance. Free exploration lets the user freely explore the representation while guidance constrains the user to a path by using forces. Guidance can vary from fully constrained guidance to forces that can be overcome, for instance, to help the user stay on or find a path. Based on previous work described in Chapter 2, notably Pokluda and Sochor [PS03, PS05] and Roberts et al. [RFC02], the different types of exploration are summarized into six methods:

- *Free exploration* enables the user to explore freely, see Figure 24(a).

- *Free exploration with guide* permits the user to move around freely, but get additional help when required (see Figure 24(b)). For example, users can be guided to a known place when they feel lost [PS05], or an attraction force can be initiated when they touch an object, so as to enable them to explore that object [KSS00b].

- *Free exploration with choice* enables the user to move around and then be instantly teleported to another place on request (see Figure 24(c)). E.g., Magnusson and Gröhn [MRG04] present a world where the user can freely



(a) Free Movement. (b) Free movement with guide. (c) Free movement with choice.

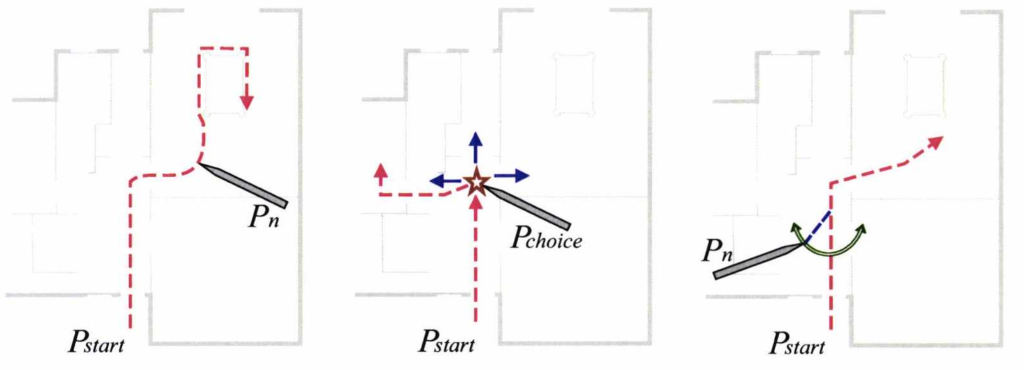
Figure 24: 24(a) Free Movement – the user can explore anywhere. 24(b) Free movement with guide – (e.g.) if lost, the user will be taken back to a known point. 24(c) Free movement with choice – the user can move around and then choose to be hyper-jumped to another place.

explore in a virtual box, but when the user pushes the walls of this box, or hit a key on the keyboard, he/she is moved to another virtual world.

- *Guided tour* which fully controls the movements of the user and shows them interesting features, Figure 25(a) [PS03, PS05, RFC02].
- *Guided tour with choice*, where the user is initially led and then given some choice of where to go next, Figure 25(b) [RFC02].
- *Guided tour with freedom* where the user is given some freedom to move as they are led along a route, Figure 25(c) [RFC02, KSS00b].

Consequently, a combination of both guidance and free exploration techniques using the different types of exploration presented above, especially with guidance first followed by free exploration, could help the user get a feeling of the different items and of their location and relative positions before letting them freely explore the items to strengthen the constructed mental model. As it is easy to get lost in non-visual 3D worlds, initially providing guidance can mean the user avoids exploring the model without finding the items and getting frustrated.

Haptic guidance has been investigated in other areas such as for handwriting applications [BPK05, TBL02] and for path navigation tasks such as in virtual mazes [BPK05]. Bayart et al. developed a “record and progressive-replay” strategy, which consists of a decreasingly constrained guidance to reduce the dependence on the teacher. The guidance range from fully constrained to partial

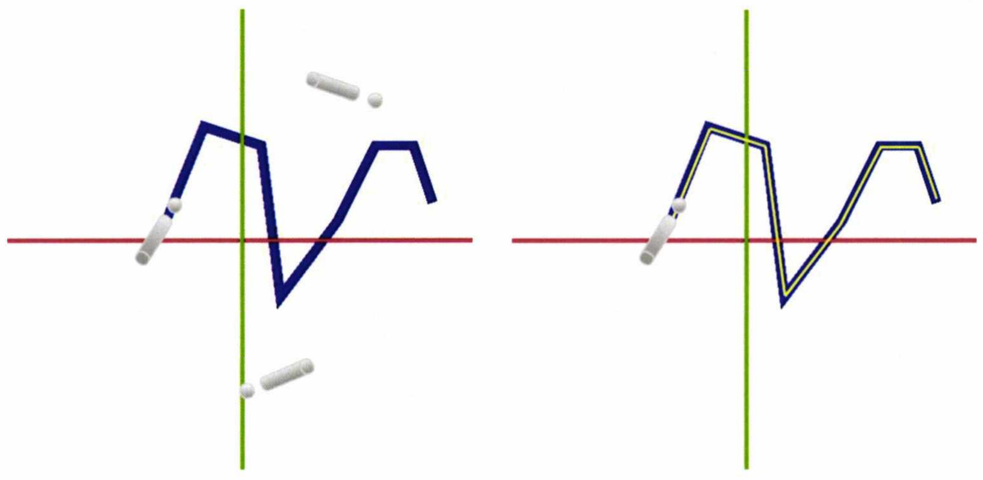


(a) Guided tour. (b) Guided tour with choice. (c) Guided tour with freedom.

Figure 25: 25(a) Guided tour – the user is led along a predefined route. 25(b) Guided tour with choice – the user is led along a predefined route and then given a choice of directions. 25(c) Guided tour with freedom – the user is allowed some movement as they are led along a predefined route.

guidance, where kinesthetic forces are applied to get back in the direction of the track, and finally to simple correction with forces that help the user stay on the line. They tested it on two applications, namely a handwriting task and finding the way out of a 3D maze. The maze application highlighted that the adaptive haptic guidance achieves the best time performance as compared to free exploration and full guidance, especially the more complex the maze. Teo et al. [TBL02] developed a robotic teaching system, coupling visual and haptic feedback. The teaching system used two guidance modes: the motion guidance mode, which is equivalent to a full guidance, but speed can be changed along with stiffness and damping to reduce the level of assistance by reducing the force constraint, and the path guidance mode where "the student is constrained to a path but free to perform the movement at any desired speed..., or even to pause during a stroke", which is equivalent to the partial path guidance from Bayart et al. explained above. They conducted two experiments with three users in each to evaluate the impact of haptic guidance on two types of writing tasks (pen writing and calligraphy) and found that in both cases, haptic guidance improved the learning with an average rate of 41%. In the case of line charts, Roberts et al. [RFC02] suggested some guidance-exploration models but did not implement these. The line chart application draws inspiration from their ideas and presents initial implementation and evaluation of these guidance-exploration models.

The main design goal is to provide a method that gives an overview, using



(a) Free Exploration (b) Magnetic Line

Figure 26: The unconstrained and partially constrained models: 26(a) The user can freely explore the model, here three different user positions are represented, that explore the axis, the line or the background; 26(b) The user can freely explore the model; however a magnetic line (yellow line) has been added to the line in order to help the user explore and stay on the line.

solely the haptic modality. In the case of a line chart representation, the overview includes conveying the line chart relative location to the axes, its overall shape and some specific features that help to build the mental model, such as maxima, minima and intersections with axes.

3.2 Design and Implementation

Five guidance-exploration models have been developed that the user can choose from using the keyboard. The following figures 26, 27 and 28 are screenshots from the prototype application, enhanced with explanations to demonstrate each model. Figure 29 depicts screenshots of the haptic setup with a user exploring the line chart.

First, the unconstrained navigation model lets the user freely explore the area on their own (see Figure 26(a)) and corresponds to the *Free exploration* model.

Second, free exploration with a magnetic attractive force that is added to the line helps the user stay on the line (see Figure 26(b)). This exploration concurs to the *Free exploration with guide*, as the user can freely explore the line chart and when required, can ask for help to stay on the line.

The last three models ('bus tour', 'water skier tour' and 'museum tour')[RFC02]

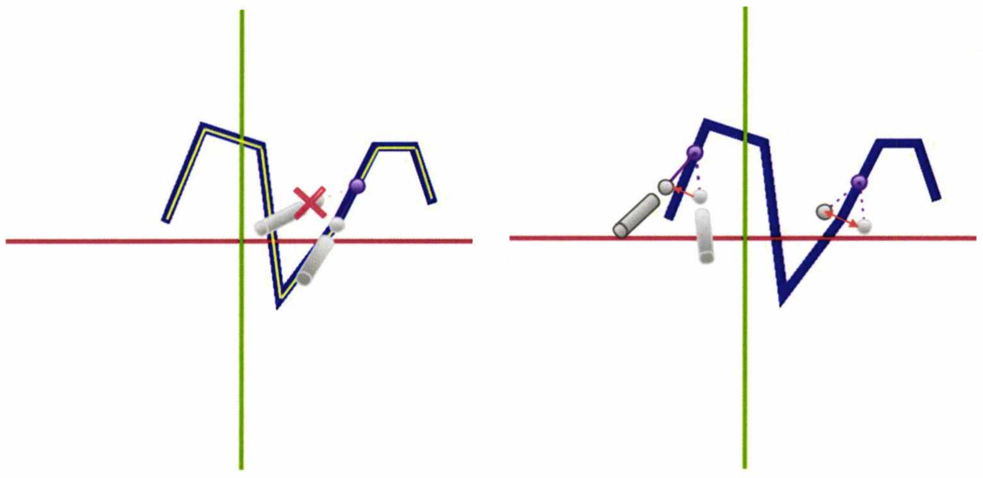

(a) Bus Tour (b) Water Skier Tour

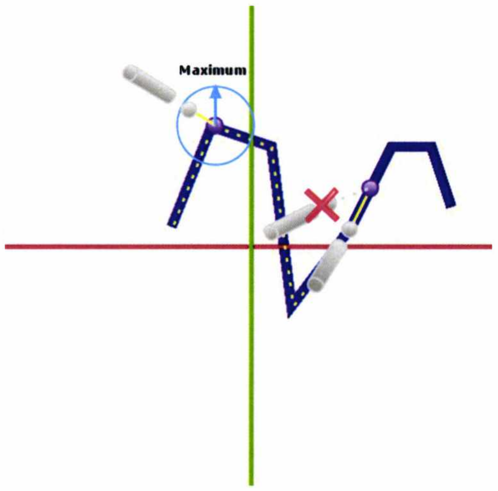
Figure 27: The bus and water skier tour. The bus tour is a fully constrained guidance model and takes the user along a predefined path (see 27(a)). A magnetic line (yellow line) is added to the guidance to ensure the user is constrained to the line. The water skier tour also takes the user along a predefined path but allows for movements sides to sides (see 27(b)).

connect the stylus through a spring force to an anchor which is moved by the program to guide the user on a predefined route. The fully constrained 'bus tour' model (see Figure 27(a)) glues the stylus directly to the line with a magnetic effect, constraining the user movement to the line only and thus is equivalent to the *Guided tour.*

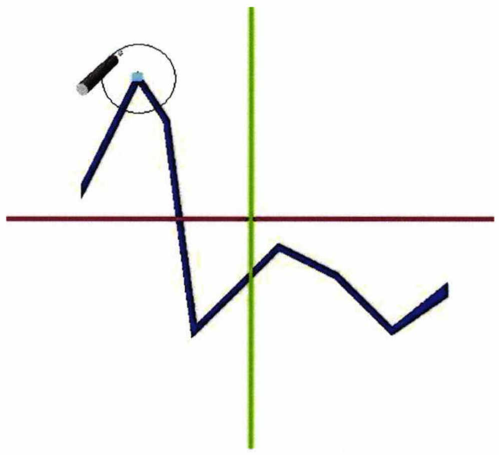
With the 'water skier tour' model, the user is able to move around while being pulled by the anchor (see Figure 27(b)), which conforms to the definition of the *Guided tour with freedom.*

Finally, the 'museum tour' is similar to the 'bus tour' except that it pauses at points of interest and lets the user freely move around within a certain range for a fixed time to explore these areas before resuming (see Figure 28). This matches the *Guided tour with choice* as the user is initially led to a place of interest and then given some freedom of movement, before being led again.

For all the guidance models, the anchor first guides the user to the origin and then along the axes to the first point where it pauses before starting the tour. This has been chosen rather than taking the user directly to the line in order to give an estimate of the first line point position and thus help locate the line chart. The speed at which the anchor moves can be changed to highlight different features. E.g., in this application, the movement of the anchor was slowed down at the

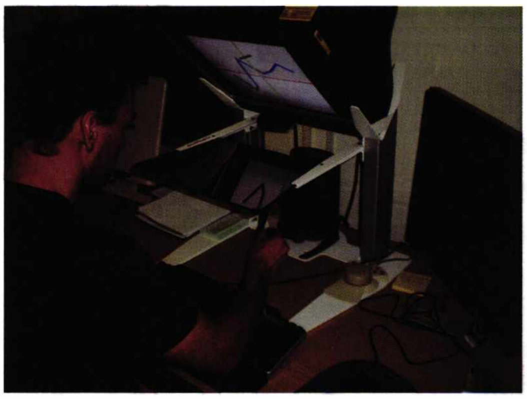


(a) Museum Tour

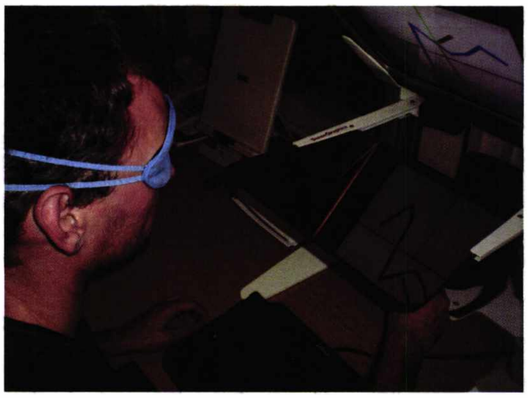


(b) Museum Tour Screenshot

Figure 28: The museum tour. 28(a) presents a schematic of the museum tour while 28(b) is a screenshot of the museum tour in the application, with the added circle to represent the device range of movement. The blue square (in 28(b), purple circle in 28(a)) is the 'anchor' that the haptic device becomes attached to. The device is led along the shown engraved line, but stops for a given time to allow for user exploration at points of interest, which are maximum, minimum points and intersection with axes. At these points, the user can freely move within a local area, represented by the circle, before the tour resumes.



(a) User with application



(b) Blindfolded user

Figure 29: Screenshots of the setup and of a blindfolded user navigating the line chart.

intersections with axes and the minimum and maximum values. This allows for a smoother stop in the case of the museum tour, while for the bus and water skier tour it gives a hint on the presence of a point of interest when passing by it.

From the categorization of active exploration modes presented above (see Figure 24 and 25), these five explorations models cover most of the active exploration modes. Other design variations are certainly possible and some are available, e.g. for the *Guided tour with freedom*, instead of using a single force leading the user onto a path, such as in the water skier tour, König et al. [KSS00b] proposed a method where small marks (attractive forces) are placed through the scene to designate the path and guide the user. However, these five explorations models were chosen and developed as an initial step in the testing of appropriate metaphors to convey the overview.

These guidance-exploration models each offer a different degree of constraint on the user's movement, from none to fully constrained. These guidance models should be used instead of a fully constrained guidance to provide a better mental picture of the chart. Indeed, Bayart et al. [BPK05] underlined that fully constrained guidance seems the least efficient when learning a path through a maze, as users learn from their mistakes. Additionally, their experiment resulted in better performance using their adaptive mode, which is a combination of guidances with varying degrees of constraint, similar to the exploration models presented above. To evaluate the benefits of these models, as compared to free exploration and fully constrained guidance alone, a pilot study was conducted, which is described in Section 3.3 (and summarized in [PR07]).

Two further design ideas were implemented and discarded. The first design idea aimed to provide the line gradient information through the line representation. The metaphor was one of the line being like a mountain to climb where the user would go up and down, depending on the gradient of the line segments. The sign of the gradient was taken to give the user a feeling of 'up' and 'down' by varying the depth of the V-shape line. When the gradient was positive, the depth would gradually decrease until the 'up' point; while when the gradient was negative, the depth would gradually decrease until the 'down' point. In the case of a straight segment, the depth would not change. However, to obtain a feeling of going up or down, a noticeable change in the depth was required, meaning that the 'down' depth had to be a quite high value. Consequently, this would cause the line to be quite deep in some places and having a steep V-shape, which would prevent the user from smoothly leaving the line without losing contact with the line and the chart in the case where the user would choose to "quit" the line and explore the background again. Although this technique was not formally tested, it was discarded as losing contact with the chart is not a suitable behaviour for non-visual exploration. Moreover, the difference in depth variation needed to be high to be noticeable and therefore troublesome to achieve to simply convey the shape of the line. Another design idea, that has not been implemented, was to vary the tour speed to provide that same information. While going 'up', the speed would be reduced whereas while going 'down' the speed would be increased. A corresponding metaphor would be of being on a rollercoaster.

The second idea was a 'push menu'. This menu idea was inspired from the four colours pen, where users change the pen colour by pushing down the corresponding colour on the top of the pen, which in turn would push up the last chosen colour. A similar menu was initially created with four choices: free exploration and the three guidance tours. The menu was circular and each quarter could be pushed down by applying a force with the haptic device. When the quarter would reach its final position, the last chosen quarter was pushed up to its initial position. A force threshold was used to indicate the difference between 'feeling' which exploration mode was chosen and 'pushing' down the quarter to change the exploration mode. The quarters were surrounded by a cylinder so that the user doesn't slip off at the edges. This push menu was used during the pre-pilot. However, this initial study clearly showed that the keyboard was much easier to use, especially as it enabled users to keep the pointer in contact with the line while changing the exploration model. Also, the implemented menu exhibited some 'fall through' problems which made it more difficult to use, i.e. where the PHANTOM pointer would fall through the haptic surface when it is not supposed to. Therefore this menu idea was also discarded.

Finally, a last idea for an exploration model was to cast some magnetic rays, at a given point line position and towards each axis, that would convey an estimate of the point coordinates by letting the user feel the position on the axis. This is similar to when a user is visually reading coordinates and projecting the point x and y coordinates on the respective axis. This could be used as a stand-alone functionality or added to the museum tour for the points of interest for instance. However, this idea has not been implemented nor tested and was left for future work.

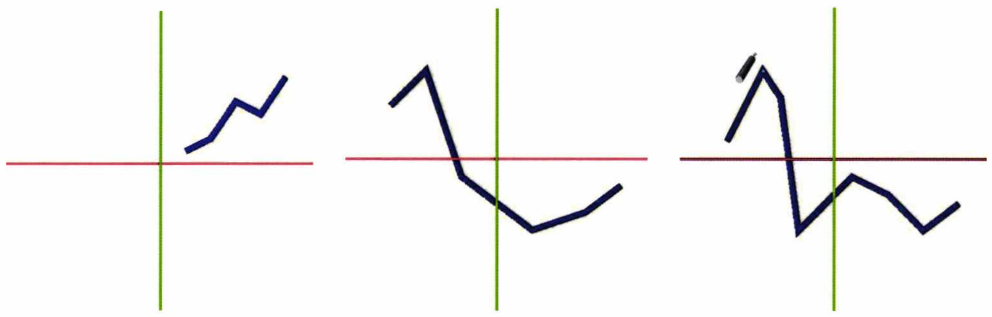
The line chart application was built with the H3DAPI^{TM} [H3D10], a haptics software development platform, and uses the force feedback device PHANTOM desktop. The prototype displays a single line chart that the user can explore. The haptic line is represented using the engraved modelling technique (see Figure 28(b)), as Yu et al.[YB03], which prevents the stylus from slipping off the line. Unlike previous work [YRB00, YCB02, YB03], the whole area (positive and negative values on axis) has been represented and some 'bumps' have been added at the intersections of the engraved line with the axes. Each axis is modelled using cylinders. Thus, when the user explores the graph outside the line, the axis is solid and they cannot pass through; whereas when the user is exploring on a line they feel a small bump as they pass over an axis. The line graph and the axes are placed on a 2D background surrounded by walls.

3.3 Evaluation and Results: Pilot Study

3.3.1 Pilot Study: Experiment Design

The pilot study was conducted with three computer science students with no or little experience with haptics, who were blindfolded during the study. This initial study was conducted to check if preliminary results support the research design goal and so the direction of the work was sensible. The hypothesis was that the guidance-exploration models would increase the amount of information gathered and thus enhance the overall understanding of the line graph compared to free exploration only (or guidance only). More specifically, the guidance-exploration models would improve the user's understanding of the interesting features (i.e. the number and locations of points of interest) and general layout. The study procedure had four parts: (1) a familiarization phase, (2) an exploration phase, (3) a drawing phase and (4) a questionnaire and interview phase. The study materials are attached in Appendix A.

During the familiarization phase, the participants were first introduced to the interface, including the haptic setup and the application guidance options. They were given three example graphs, one for each level of complexity, so they would become familiar with the line chart representation and guidance interactions. The line graphs were divided into three levels of complexity namely easy, medium and challenging depending on the total number of bends and of points of interest (see Figure 30 for an example of chart for each level of complexity). Easy graphs had a complexity equal to six or less; medium ones had a complexity between seven and nine while challenging ones had a complexity equal to more than 10.



(a) Easy Chart (b) Medium Chart (c) Challenging Chart

Figure 30: Some examples of line charts for each level of complexity: 30(a) easy (graph complexity (gc) = 5 in this case), 30(b) medium (gc = 8) and 30(c) challenging (gc = 11).

After the training phase, the participants were blindfolded and asked to explore the graphs (see Figure 29). Nine graphs were given to each of the participants, with the different levels of complexity, in randomized order. Three conditions were evaluated: (F) free exploration, (G) constrained guidance with the 'bus tour' repeated and (F+G) combined free exploration and guidance (the participant could use the 'museum tour' and/or the 'water skier tour' and/or the help to stay on the line). The participants were given 2mins for easy graphs, 2min30 for medium ones and 3min30 for challenging graphs. During the trial, they were asked to remember the points of interest (the minimum and maximum values and the intersections with an axis), which they were asked to count and locate on the line chart during the drawing phase at the end of each trial. They were also asked to draw the overall shape of the line. At the end of the experiment, they were asked to answer a few questions about their preferences for the exploration interactions.

The duration of exploration for each level of complexity was chosen from the results of a pre-pilot study. The pre-pilot study was conducted with exactly the same tasks with three computer science postgraduates, and in addition to the accuracy of answers, the time was also measured. Each trial would end when the participants thought they had all the required answers. For each level of complexity, the shortest time to complete a trial was taken for each user and the maximum value was rounded to the nearest 30s. For instance, the maximum value for the easy condition was 1min50, which in turn was rounded to 2mins; for the medium condition it was 2min44 rounded to 2min30, and lastly for the challenging condition it was 3min18 rounded to 3min30. As participants were left the choice of when to end the trial, with the accuracy of answers as a priority rather than speed, the average of the times values was not appropriate as the measure for the duration for each level of complexity. Indeed, some participants took much longer than they really required. Even after knowing the right answers, they would explore longer to confirm they were right. Consequently, the longest times approached 7mins. As the accuracy would usually be quite good (3 at least out of 4), the minimum time to reach good accuracy was thus chosen instead.

3.3.2 Pilot Study: Results

The data collected consisted of the 27 drawings completed by each participant for each graph (3x9) and the participant's answers to the questionnaire. The data from the drawings was analyzed using a scoring system. The participants were asked to count the number of minima, maxima, intersections with x-axis and intersections with y-axis. For each of these categories, if the participants found the right count, they would score one point. The maximum score for a graph, corresponding to the right count for each category, would thus be four. The participants were also asked to sketch the overall shape of the line chart and locate the points of interest on that chart. If the points of interest were well located on the chart, a '+' would be adjoined to the score, whereas if the location was wrong for any of the points of interest, a '-' would be adjoined. These scores were grouped by the exploration condition (F, G and F+G) and the level of complexity of the line chart (easy, medium, challenging). The results are summarized in Table 1.

	Easy			Medium			Challenging		
User	1	2	3	1	2	3	1	2	3
F	2+	3+	4+	4+	4+	4+	3-	3+	2-
G	2-	4+	3-	3-	4-	3+	1+	4-	3+
F+G	4+	4+	4+	3+	4+	4+	4+	4+	4+

Table 1: Pilot study results: the participant scores 1 point for each right number of point of interest found (minimum, maximum, intersection with x-axis, intersection with y-axis); the minus indicates right answers but mislocated while the plus is for right well located answers on the drawing.

As can be seen from Table 1, the participants achieved 80.5% of correct answers for the free exploration condition, 75% for the fully guided constraint and 97.2% in the case of the combined mode. The combined mode appears to improve the understanding of the graph, particularly for easy and challenging graphs (100%

correct answers). The number of subjects is too small to consider these results significant and thus to draw any conclusions about the hypothesis tested. However, these results are encouraging and do support the hypothesis.

Additionally, during the interview, all participants commented that the guidance tours were effective and the axes and line modelling was useful, reaching score averages of 4 and 4.5 out of 5 respectively (see Table 2). The preferences for each tour varies among individuals though; the museum tour was the most used and obtained the highest average score (3.67), followed by the water skier tour rated by two users (3.5) and finally the 'help on the line' through the use of a magnetic line, which was rated by only one user and quite poorly (1). One of the participants commented preferring the water skier tour because it was more continuous and thus it was easier to get the "big picture", as compared to the museum tour which stopped at points of interest.

	Effectiveness of tours	Effectiveness of each tour			Perceived difficulty	Keyboard workload	Usefulness of axes & engraved lines
		MT	WT	ML			
User 1	3	2	5	1	2.5	5	5
User 2	5	5	NA	NA	4	3	4
User 3	4	4	2	NA	4	NA	4.5
Total (/15)	12	11	7	1	10.5	8	13.5
Average	4	3.67	3.5	1	3.5	4	4.5

Table 2: Pilot study questionnaire results. As some answers have not been answered, the average is the sum of the scores divided by the number of participants who attributed a score. 'MT' refers to 'Museum Tour', 'WT' to 'Water skier Tour' and 'ML' to the help on the line provided by the added magnetic line.

The participants also commented that they preferred the combined free exploration and guidance mode rather than free exploration alone. For instance one of the participants stated that "the guided tour combination with free tracing was good, or rather better than the 'lone' ones; I could count the mins and maxs and be sure I was right". Some of the issues reported during free exploration alone concerned how easy it was to get lost in the space, as underlined by one of the participants who commented that "the axes are difficult to calibrate with free exploration as I always went underneath", and the lack of 'continuity' as highlighted by one of the participants who said that "most of the time [I could easily locate the points of interest], except for the lone-free exploration when I couldn't really count and compare the mins and maxes". Summarized feedback from the usability questionnaire is included below.

1. Rate the following areas (see Table 2 for a summary of the score results):
    - Overall effectiveness of the guided tours (1 not effective at all, useless - 5 very effective very useful)
    - Effectiveness of each of the tours you have tried, namely Museum tour, Water Skier and Magnetic line (same scale as before)
    - Perceived difficulty of the task (1 very hard - 5 very easy)
    - Workload of the keyboard use/tours associated to keyboard (1 very high workload - 5 very low workload)
    - Usefulness of axes, engraved line (1 totally useless - 5 very useful)
2. Cite drawbacks:
   "*Remembering all lines of complex graphs.*"
   "*Only in the training, I felt the graph lines should have been deeper; but for the later testing there were deeper so it was ok.*"
   "*The hand gets weary without support. The axes are difficult to calibrate with free exploration as I always went underneath.*"
3. Cite good points:
   "*General shapes were very easy to remember.*"
   "*The guided tour combination with free tracing was good, or rather better than the 'lone' ones; I could count the mins and maxs and be sure I was right.*"
   "*Axes easy to notice and the troughs and high points easy as well.*"
4. Was it easy to navigate?
   "*Yes easy - small differences in slope are hard.*"
   "*Some time for harder or more difficult graphs I kept falling out of the graph lines and then back on.*"
   "*Yes*"

5. Was it easy to locate specific points of interest?
   "*Yes - but which one/type it is, is harder.*"
   "*Most of the time, except for the lone-free exploration when I couldn't really count and compare the mins and maxes.*"
   "*Yes*"

6. Are there some elements you would have liked to be available in the interface?
   "*Easier distinguishing of start + finish.*"
   "*I think it was quite nice and nicely presented.*"
   "*Sound when getting to interesting places.*"

7. Were the instructions easy to understand?
   "*More explanation of different navigation modes maybe useful.*"
   "*Pretty much.*"
   "*A bit ok.*"

8. Additional comments:
   "*Overall I think it was a good test with instructions presented beforehand and help through the training.*"

Three other problems were reported during the interview: (1) the too sudden stopping of the guidance; (2) the slipping off the line at sharp bends and (3) the non-tunable speed. The first problem arised when the guidance stopped and the anchor was removed (see Figure 31); it appeared to be quite sudden and tended to make the participants slip off the line. One participant mistook this slip off for a bump and thus as an intersection with an axis. Hence, that participant suggested that the start and end point of the line should be made easier to distinguish. Falling out of the line chart would also happen during the exploration of challenging graphs. One participant commented that "sometimes for harder or more difficult graphs I kept falling out of the graph lines and then back on". Indeed these graphs contained many bends and often sharp ones; the participants would be behind the anchor, reaching for the tip of the bend when the anchor would already be moving forward on the other side of the edge, thus pulling the participant strongly toward itself (see Figure 31). This effect would also confuse the participants with intersections by giving them the feeling of going over a bump. Another participant also said that sometimes the speed of the guidance was too slow and it should be user-tunable instead.

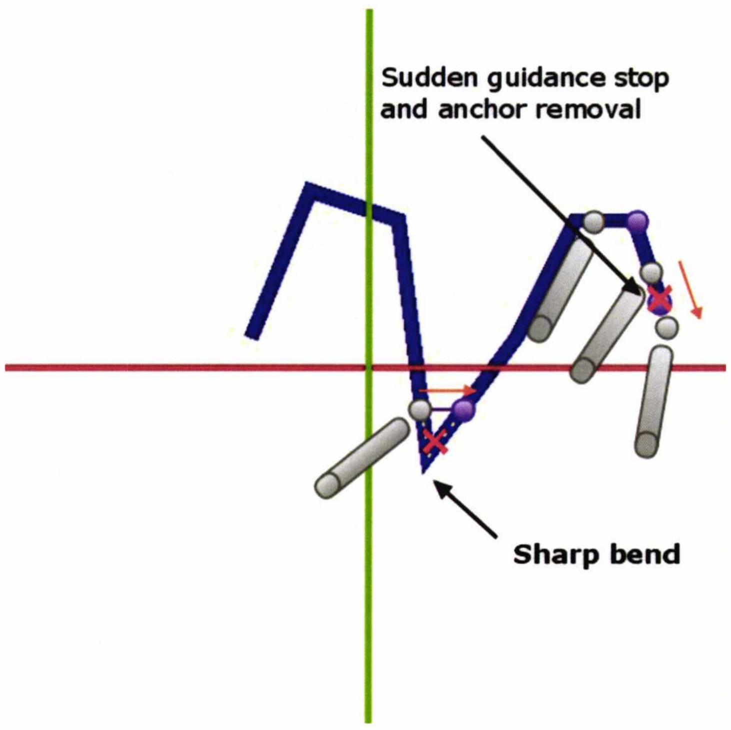


Figure 31: Two problems were highlighted by users during guidance tours: the sudden stopping of guidance and slipping off the line at sharp bends problems. These problems often caused the participants to be 'jumped' to a location (represented by the orange arrow), giving them the wrong impression of feeling a bump and thus an intersection with an axis.

3.3.3 Pilot Study: Limitations

The pilot study is the first step towards a more substantial quantitative study. Therefore, its results can be used to refine the study design and as initial indications of the possible study outcomes. As the number of participants is small and the further study quantitative, no quantitative conclusions can be drawn from the pilot's results. However, the pilot's results allow, even with a small number of participants, the discovery of usability issues, which in turn helps improving the application and the future study.

The pilot study was conducted with three blindfolded computer science students who are not representative of the target user group. Visually impaired participants are difficult to recruit. Therefore, evaluations are usually performed with blindfolded sighted participants in a first stage, to validate the design, as can be seen from Chapter 2, before testing with a few visually impaired users. The same procedure is here adopted. However, for the future evaluation, participants with a wider range of background will be recruited to avoid any bias.

3.4 Discussion

The initial results are encouraging as they seem to support that free exploration combined with guidance-models improved the understanding of the important graph features. This hypothesis is supported by the participants's preferences and, although the number of participants was not sufficient to consider the quantitative results significant, the results do support the need for further research in the area. In order to validate the hypothesis though, more participants should be recruited. However as previous studies in other areas proved that a guidance-model, where the level of movement constraint varies, improves the performance over free exploration alone [BPK05], we believe it would also be the case for the line chart application.

It is also envisaged that the exploration techniques employed here would be useful to distinguish multiple lines. Guidance explorations that take the user along each line and give information about intersections with other lines, could give the user a general idea of the lines shapes, their relative locations to each other as well as their intersections, which could be then confirmed during free exploration.

This pilot study also revealed a few usability issues with the design of the line chart application. First, some participants suggested that a better distinction between the start/end point line and the start/end of the guidance tours should be provided. Some participants did not clearly understand when the line started as they were taken to the first line point through the origin and following the axes first. This was done to provide an estimate of the first line point location by pausing before the tour began. However, pausing at the first point was not a good cue, especially in cases where the first point was also a point of interest. On the other hand, when the guidance tour reached the end of the line, the anchor would be removed and the 'vanishing' of the pulling force would feel too sudden for some participants, giving them the feeling of a 'bump' and thus making them think the line would end at an intersection with an axis (see Figure 31). An additional cue indicating the end of a line is therefore needed. Some participants suggested the use of audio cues; but as a haptic-only alternative is being investigated, perhaps reducing the force exerted by the spring attaching the user to the anchor instead of completely removing it as soon as the guidance stops could be enough.

Secondly, some participants complained about the lack of control of the guidance tour speed. Indeed, after using the guidance tours and exploring the graph a few times, the participants wanted to be able to speed up the tour and slow down the overall speed at areas where they were in doubt. This comment stemmed mostly from the pre-pilot where the participants did not have time restrictions and thus used the tours many more times (the longest trial lasted up to 7 minutes) than during the pilot study where there were used only a few times. This has been implemented after the pilot and through the use of the keyboard (minus and plus on the numeric keypad), the speed can now be adjusted during the guidance itself.

Thirdly, it was noted by almost all the participants that sharp edges induced confusion during guidance. Indeed, the way the guidance was implemented, the participant would be following the guiding anchor, by being slightly behind it. However at a sharp bend, the anchor would already be on the other side of the bend, pulling the participant still on the first side, on the way to the tip. This would cause the participant to 'jump' towards the anchor, thus feeling a bump instead of a maximum or minimum (see Figure 31). This would often cause minimum or maximum points to be confused for axes intersections and be generally disturbing. Subsequently, a solution to this problem has been implemented in the form of new engraved modellings: a V-shape engraved line which is flattened at the lower edge for sharp bends and a U-shape engraved line. However, these improvements have not been tested with users.

3.5 Summary

The main goal of this line chart application was to investigate navigation metaphors to solve the issue of getting an overview haptically. To that effect, five exploration models were implemented and tested with a few users. During the evaluation, participants preferred the combined guidance and free exploration, with different preferences for the individual guidance interactions. Although the pilot study did not have enough participants to draw conclusions, preliminary results seem to point towards the validation of the hypothesis that the combined mode can improve the mental picture of the line chart, as demonstrated in previous research for other application areas [BPK05].

During the development of the line chart application, two main issues, which significantly slowed down the achievement of the main goal of the application, i.e. the investigation of interaction metaphors, were encountered.

- *Implementation*: to generate and explore the different models, new code was required. Implementing the different models took many hours of coding, which was divided between researching how to develop the idea using the API and then programming it until the desired effect is reached, while dealing with the API limitations. Consequently, it was difficult to explore different scenarios and quickly adapt the code, before being able to test them with participants.

- *Testing*: evaluation is an important step, required to validate a design and upgrading it from the mere idea to an approved or discarded result. Testing can also lead to suggested improvements and consequently stimulate new ideas for improved or new designs. Testing goes hand-in-hand with implementation as the development process is usually iterative. Therefore, slow implementation hinders the rapid testing of designs. Testing should happen as early as possible, to discard unsuccessful ideas and be more efficient in the development of designs. This is difficult with slow implementation, which delays the testing phases. Similarly, the benefits of testing are reduced if the improvements or new ideas take time to be implemented.

These issues generally restrain the investigation of navigation metaphors and particularly finding, developing and testing new interactions quickly. Moreover, they make investigating new interactions challenging for programmers, but inaccessible to designers or teachers for visually impaired people, with no programming knowledge, without the recourse to a programmer. However, in many cases, designers or teachers have a clearer idea of the needs of the target audience and enabling them to create designs would surely foster the investigation of effective interactions.

Therefore, researchers have proposed solutions to facilitate the development of designs and interactions, and in particular haptic interactions. The two main solutions consist of firstly, providing guidelines and methodologies to ensure the efficiency of the design and implementation process, and secondly, providing the ability to develop and test rapidly haptic interactions, through prototyping languages or tools. These solutions will be further explored in the second part of this thesis and in particular led to the development and evaluation of a prototyping tool.

Part II

Facilitating the Development of Haptic Interactions

Chapter 4

Methods to Facilitate the Development of Haptic Interactions

One of the principal challenges outlined in Chapter 2 was that of interactions, and in particular navigation. That is, users find it difficult to interact with the haptic visualization, in particular to know where to move the haptic device such to feel the data. Interactions are a key component in providing an understanding of the underlying data and therefore, developing effective interactions is a major concern as it can help the user in their discovery. However, as highlighted at the end of Chapter 3, developing effective interactions can be challenging and limited to people with technical knowledge.

Various approaches have been proposed to overcome these difficulties, and so facilitate the design of haptic interactions and in turn promote the design and development of suitable solutions to help the user interact and navigate in haptic worlds. The approaches can be divided into two main methodologies: taxonomies and frameworks, which aim to structure the interaction space and the interaction development and thus guide the development; and prototyping languages, which enable rapid development and testing of various designs.

This chapter explores these different methodologies, and in particular those that apply to haptic interactions. First, the challenges for the development of haptic interactions are detailed in Section 4.1. Subsequently, some definitions are given about interactions techniques in Section 4.2 to help understand concepts described through the various taxonomies and frameworks that classify the interaction techniques and their development process in Section 4.3. Finally, Section 4.4 presents different prototyping languages.

4.1 Challenges

Designing interactions for 3D virtual environments presents many challenges, especially in non-visual applications. Even though humans naturally interact with a 3D world in their everyday life, users still have difficulty understanding the cues of a 3D virtual world and thus can have difficulty interacting and moving in it. Users often lose their spatial orientation and location within the environment or have trouble using the available interaction techniques. Indeed, 3D virtual worlds are still missing many cues present in real environments and often the available 3D interaction devices are limited and do not offer natural interactions (one point interaction for example, limited freedom of movement). These challenges are accentuated in non-visual 3D virtual worlds, where the modalities have lower bandwidth than vision and where the devices have additional limitations.

In the virtual reality domain, considerable research has been conducted to improve the quality of the design of 3D interaction techniques and thus the quality of the experience in the virtual world. Most of the interaction techniques have been tailored to a specific application and so have not been evaluated against each other [BH99] which, alongside to the challenges of 3D virtual space, have motivated the formalization of their design. Researchers have investigated different approaches to help developers efficiently design their application's interactions. One approach commonly adopted is to build taxonomies that include detailed classifications, guidelines, rules and/or processes to give the developer a methodology, founded on previous findings. The other common approach consists of defining a high-level language that the developer can use to quickly design and test interactions, that can also be referred to as prototyping languages. These approaches will be further explained in the following sections.

4.2 Interactions Techniques Background

An interaction technique (IT) is a method that enables users to carry out a task in a virtual environment through the use of interaction devices. Consequently, a haptic interaction technique (HIT) is an interaction technique that uses haptic feedback [LD04].

Interaction techniques are often classified according to the type of tasks they support. For instance, Jesper Kjeldskov [Kje00], divided interactions tasks into: orientation, navigation and manipulation that he later renamed as orientating, moving and acting tasks [Kje01]. Another common classification groups the tasks into selection and manipulation of objects, navigation, and system control, or selection, manipulation and travel [BH99] where system control tasks can "generally be characterized as selection and/or manipulation tasks".

Although the classifications of the main types of tasks slightly differ, they share common definitions. These definitions are summarized here in order to introduce the concepts and structure used by the taxonomies presented in the following section.

Navigation : usually refers to "being able to move/find one's way through the virtual world" [Kje00]. This definition includes two main aspects: 'moving' and 'finding one's way' or 'moving from the current location to the desired point' and 'finding and setting routes to get to a travel goal', which can be considered as falling under travelling and wayfinding tasks respectively [BKH97]. Bowman and Hodges [BH99] state that "Travel is part of the larger task of navigation, which includes both the actual movement and the decision process involved in determining the desired direction and target of travel (wayfinding)".

Selection : refers to "the picking of one or more virtual objects for some purpose" [BH99], such as highlighting objects to perform other actions on them.

Manipulation : involves "positioning and orienting" [BH99] objects or more generally acting on an object. For instance, the deformation of an object is a manipulation task. Manipulation tasks are often coupled with selection tasks.

System Control : these tasks encompass the commands issued to control the application. For example, to allow actions to be performed at a system level (saving/loading files or objects) or to activate some functionality (e.g. activate some view modes) [BH99].

Interestingly, when categorizing haptic interaction techniques, Lécuyer and Dominjon [LD04] added two more task categories to Bowman and Hodges' [BH99] classification, namely visualization (information extraction) and communication (information transmission). However, these latter categories do not really fit as atomic tasks of a virtual application but rather as higher level tasks or goals that involve a combination of navigation, selection and manipulation tasks to convey the underlying data or the sense of presence for collaborative applications.

4.3 Interaction Taxonomies

This section describes the background research in supporting the design of user interactions by formalizing their development through taxonomies and guidelines. This research is divided into the following subsections: virtual reality, haptics and visualization.

4.3.1 Virtual Reality

Mine [Min95] categorized virtual interaction techniques to help the developer design them in a more natural and intuitive fashion and encourage the creation of new ones. He divided interactions into five classes: movement, selection, manipulation, scaling and virtual menu and widget interaction. He also listed the three main categories the interaction techniques can be grouped into to specify the parameters of the associated tasks (direct user interaction, physical controls and virtual controls) with their drawbacks and advantages. His resulting taxonomy for movement, selection and manipulation (see respectively Figures 32, 33 and 34) tasks is similar to the later taxonomy proposed by Bowman and Hodges [BH99].

Bowman and Hodges [BH99] elaborated a methodology, focusing on the design, evaluation and application of interaction techniques for complex immersive virtual environments (VEs) in order to improve the usability of such applications. Their methodology (see Figure 35) not only relies on a detailed taxonomy of the interaction tasks (defined on three levels with task, subtask and technique components, see Figures 36 and 37) where "An interaction technique is made up of one technique component from each of the lowest-level subtasks", but also is concerned with the influence of outside factors (tasks, environment, user and system characteristics) and the use of multiple performance measures "that cover a wide range of application and user requirements" [BH99] (such as speed, accuracy or completion time). They conducted several experiments to evaluate and rate different types of interaction techniques for a specific task using their methodology.

However, they did not evaluate their methodology according to the display

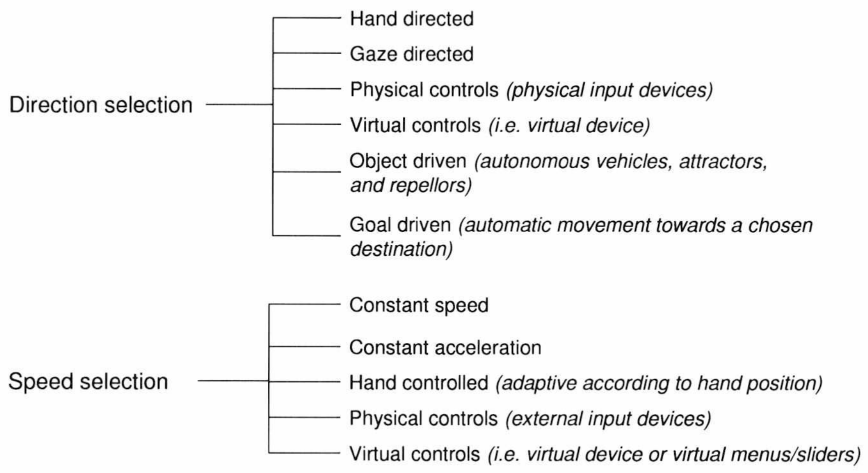

Direction selection
Hand directed
Gaze directed
Physical controls (physical input devices)
Virtual controls (i.e. virtual device)
Object driven (autonomous vehicles, attractors, and repellors)
Goal driven (automatic movement towards a chosen destination)
Speed selection
Constant speed
Constant acceleration
Hand controlled (adaptive according to hand position)
Physical controls (external input devices)
Virtual controls (i.e. virtual device or virtual menus/sliders)

Figure 32: Summary of the movement task taxonomy proposed by Mine [Min95].

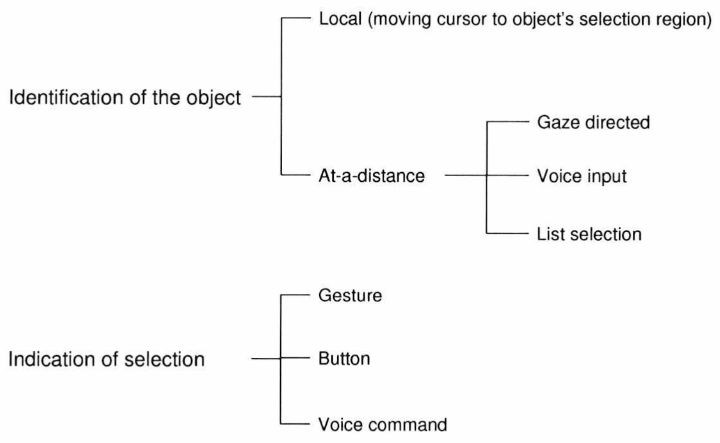


Figure 33: Summary of the selection task taxonomy proposed by Mine [Min95].

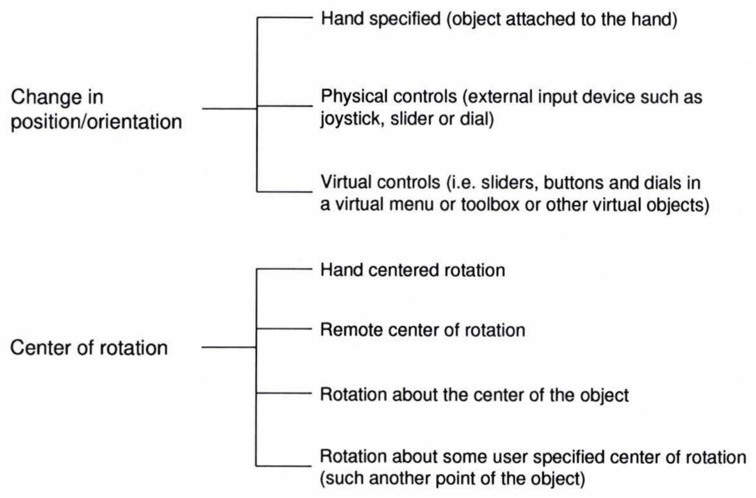

Change in position/orientation
Hand specified (object attached to the hand)
Physical controls (external input device such as joystick, slider or dial)
Virtual controls (i.e. sliders, buttons and dials in a virtual menu or toolbox or other virtual objects)
Center of rotation
Hand centered rotation
Remote center of rotation
Rotation about the center of the object
Rotation about some user specified center of rotation (such another point of the object)

Figure 34: Summary of the manipulation task taxonomy proposed by Mine [Min95].

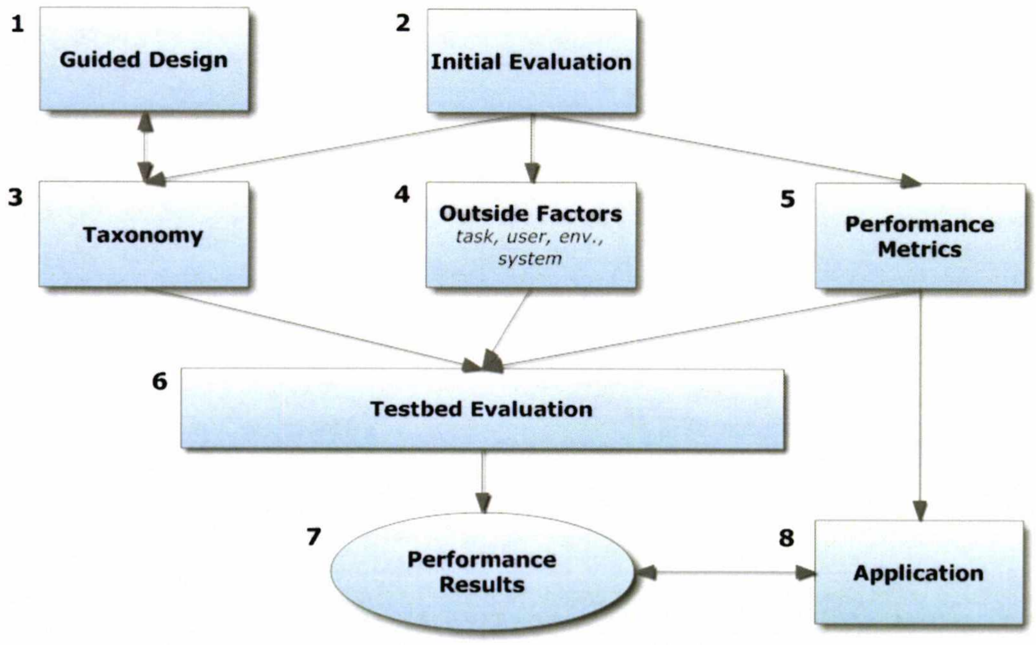


Figure 35: Methodology for the design, evaluation and application of interaction techniques elaborated by Bowman and Hodges [BH99].

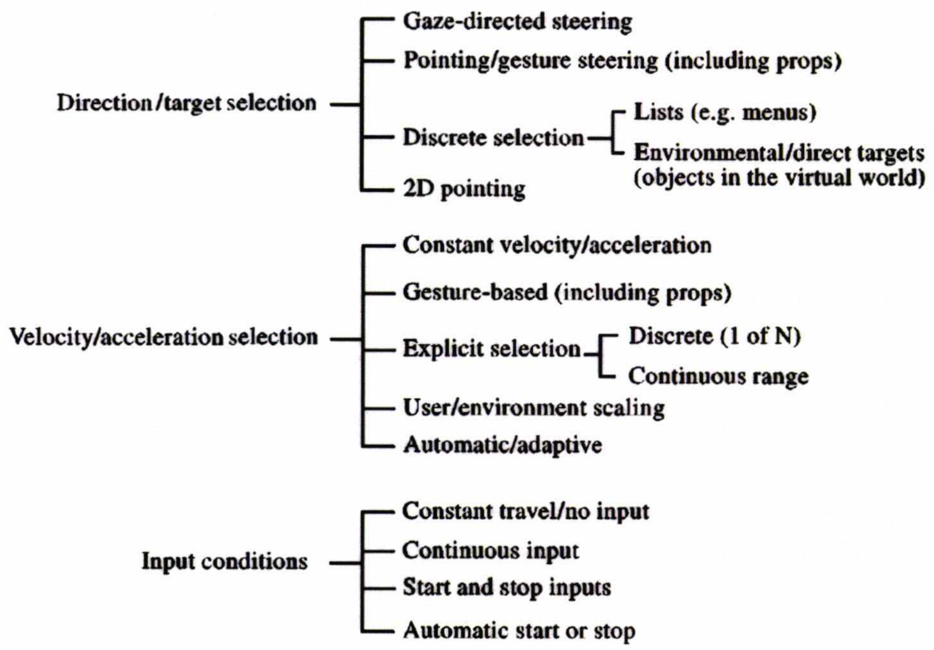


Figure 36: Travel Taxonomy described by Bowman and Hodges [BH99].

type (fully or partially immersive); thus Kjeldskov [Kje01] designed a framework with the different types of interaction techniques/devices and the display types. He evaluated more than 40 interaction techniques with the different combinations of interaction techniques/display type to highlight that the display type influences the performance of an interaction technique and that they are not equally suitable for fully and partially immersive environments.

These taxonomies and frameworks are useful for categorizing interaction techniques and for helping evaluate their effectiveness according to the type of task and/or the display types. Although this work is within the context of virtual environments, where the focus is on the sense of presence and the user's experience in the world rather than on conveying underlying data, the taxonomies described and the design and evaluation frameworks developed can be extended and applied to the haptic visualization context. In particular, these taxonomies and development methodologies can be used during the design and development of haptic interactions, as the taxonomies are independent of the interaction modality and apply well to the haptic domain.

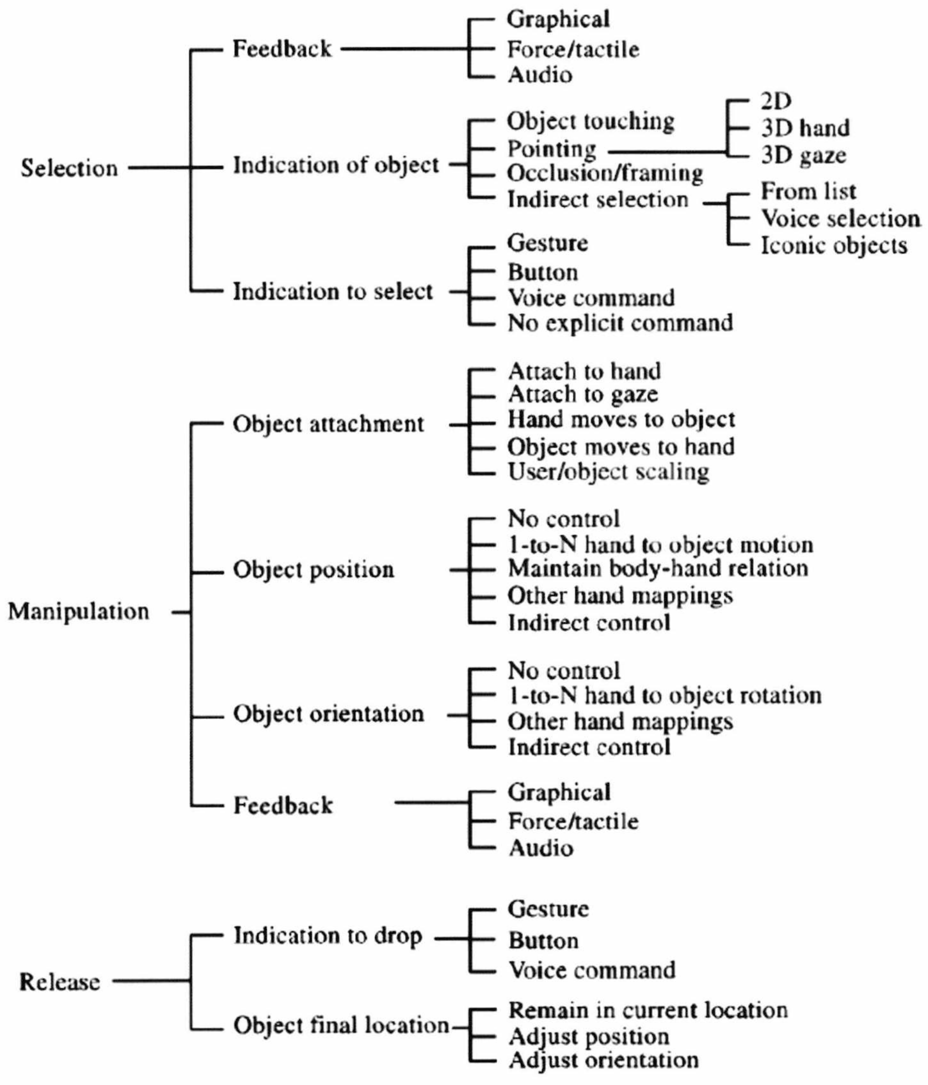


Figure 37: Selection and manipulation taxonomy proposed by Bowman and Hodges [BH99].

4.3.2 Haptics

Related work in Haptics ranges from initial guidelines and rules to a more comprehensive work on trying to find standards to design multimodal applications and, in particular, design haptic interactions. Oviatt [Ovi99] tried to demystify the knowledge about multimodal interaction through empirical evidence to provide "a better foundation for guiding the design of future multimodal systems". She lists 10 myths that she explains and disproves including the myths that "all users multimodal commands are integrated in a uniform way", that "if you build a multimodal system, users will interact multimodally" or that "different input modes are capable of transmitting comparable content". Although these myths were investigated with a focus on speech input and without mentioning any forms of haptic feedback, they still apply to applications integrating haptics and can be used as guidance. Hale and Stanney [HS04] summarize the mechanoreceptor characteristics for the tactile and kinesthetic sense and derive psychophysical tactile, kinesthetic and multimodal interaction design guidelines. For example, the kinesthetic guidelines suggest to "Add kinesthetic information to enhance objects' spatial location", to "Avoid precise motion gestures, as making accurate or repeatable gestures with no tactile feedback is difficult" and that "Gestures should be intuitive and simple". Van Erp [vE02] provides guidelines for active tactile displays from neurophysiological and psychophysical data. The guidelines are grouped in the following four categories: the detection of a stimulus, the discriminability of stimuli (or possibilities of information coding), issues related to comfort and possible pitfalls in the application of (multiple) tactile stimuli.

In the domain of 'non-visual' applications (often motivated by accessibility research), Colwell et al. [CPK+98] conducted three studies exploring the perception of virtual textures and objects, and compared the results for blind and sighted individuals. They summarized their results into guidelines for virtual textures, virtual objects, complex objects and the haptic space and its navigation. Although these guidelines primarily concern the haptic representation, they can be useful in the design of haptic interactions. Similarly, Challis [Cha00] (see Chapter 9) concluded his thesis with some interesting design principles addressing the general layout, tactile symbols, interaction and non-visual multimodality. Even though these principles aim primarily at "the design of static displays to be used as part of a computer-based system", most of these guidelines are general and apply to any haptic application. He recommends, for instance, avoiding direct translation from the visual to the haptic display, but keeping a semantic translation with the actions being consistent from one display to another. Moreover, 'empty spaces' should be excluded to avoid the user getting lost in the haptic space. Finally, Sjöström [Sjö01a] presented some rules of thumb of point interaction haptics to serve as "general guidelines for all developers of haptic interfaces for blind people". He extracted these guidelines from his work on haptic applications and interactions for visually impaired people. These guidelines include: navigation (use of reference points, stable reference systems), finding objects and overview (large objects vs thin/small, use of grooves, magnetic lines, ridges, search tools), understanding objects (help to follow the outline of the objects, avoid sharp edges and corners), haptic widgets and physical interaction (consider the design of the manipulandum of the haptic device). He later refined these guidelines in his PhD [Sjö02] and grouped them into the following categories: elaborate a virtual object design of its own, facilitate navigation and overview, provide contextual information, utilize all available modalities and support the user in learning the interaction method and the specific environments and programs (see Chapter 7).

Miller and Zeleznik [MZ99] produced a set of haptic principles to help the development and evaluation of haptic interfaces. They consider the case where "... a task to be performed by the user is given, and [they] want to choose what to simulate to allow and aid performing that task". Their "taxonomy of haptic feedback" groups the uses in four main categories: anticipation (force to indicate the imminence of a change), follow-through (feedback to indicate a change happened), indication (feedback, possibly quantitative, about an on-going action), and guidance (movement constraint); as well as a specific usage of anticipation and guidance (for distinguishing directions). These categories could be regrouped into two main categories: force used to provide qualitative and/or quantitative information about the action status (starting, on-going, achieved, etc.) and to provide some control over the action (free exploration or guidance). They also illustrate their taxonomy in the context of designing 3D haptic widgets for polygonal modelling applications, where a widget seems to refer both to representation effects and interaction techniques. Although they designed these principles in the context of haptics used to enhance visual applications, these principles also apply to non-visual contexts.

Kirkpatrick and Douglas [KD02] presented a taxonomy structured around distinct user goals in order to facilitate the development of interface evaluation techniques. In particular, they formulate the 'link' between applications and devices,

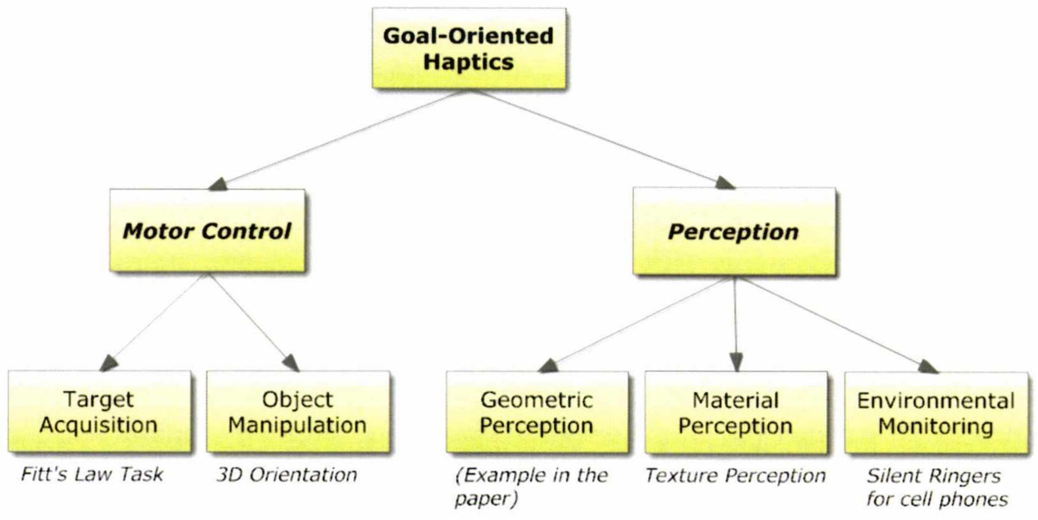


Figure 38: Taxonomy of haptic modes, after Kirkpatrick and Douglas [KD02].

by providing a taxonomy that helps evaluate whether a device is suited to an application. They explain that "metrics for interaction techniques must be defined in terms of percepts meaningful to the human user. This implies that these metrics will be determined by the application, since both the interaction technique and the percept are potentially application-specific" [KD02]. They use haptic modes to define the taxonomy (see Figure 38), where a haptic mode is "a distinct style of using the haptic system, characterized by the nature of the user's attention, the path and duration of any movement, and the skin location contacting the object of interest" [KD02]. They distinguish the goals of learning the properties of an object, such as shape, material properties (*Perception* level); and the goals of modifying the object's location (or reaching it) or structure (*Motor Control* level), see Figure 38. They conducted an experiment, to demonstrate how their taxonomy can assist evaluation. This evaluated the support of the geometric mode by point-based force feedback devices. The results were intended to be used as a benchmark.

De Boeck et al. [DBRC05] explored the existing interactions in virtual reality and their support for haptics. They first described a taxonomy where they further classified the 'metaphors' or interaction techniques for navigation, selection and manipulation tasks (see Figure 39). Navigation interaction techniques are subdivided into indirect camera control (camera moved by activating a command) and direct camera control (controlled directly by the user), which is subsequently

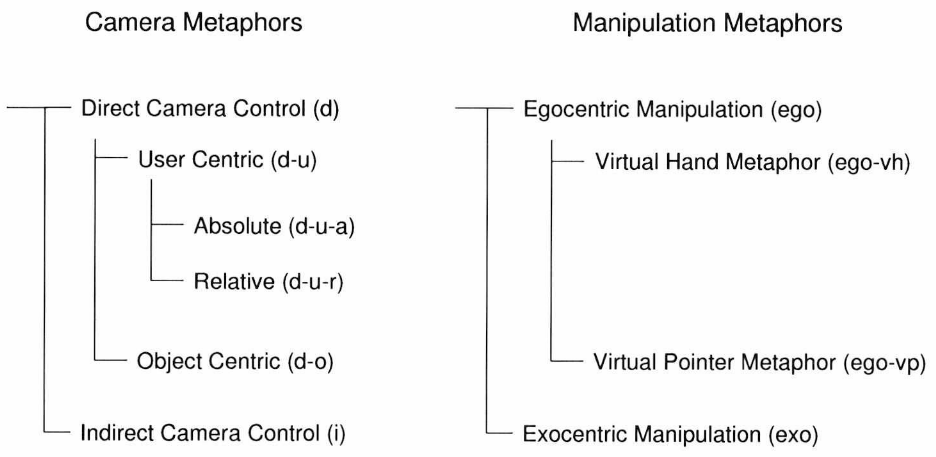

Camera Metaphors
Direct Camera Control (d)
User Centric (d-u)
Absolute (d-u-a)
Relative (d-u-r)
Object Centric (d-o)
Indirect Camera Control (i)
Manipulation Metaphors
Egocentric Manipulation (ego)
Virtual Hand Metaphor (ego-vh)
Virtual Pointer Metaphor (ego-vp)
Exocentric Manipulation (exo)

Figure 39: Summary of the taxonomy described by De Boeck et al. [DBRC05].

divided into object centric (object exploration) and user centric (scene exploration, which can be absolute, relative or both) metaphors. As for manipulation techniques, they can be classified into exocentric manipulation (action from outside the world) or egocentric manipulation (action from within), which is further divided into virtual hand metaphors and virtual pointer metaphors. They use this taxonomy to categorize existing known interaction techniques from virtual reality and study whether they can include haptics. Their results for navigation interactions are summarized in Table 3.

As haptics is a relatively new area when compared to graphics or audio, there is a major interest in defining guidelines and standards for designing haptic applications. The GOTHI model aims to "organize guidance on and development of various tactile and haptic interactions" mostly for designers [CFF$^+$05]. The taxonomy is divided into: tactile/haptic inputs, outputs, and/or combinations, attributes of tactile/haptic encoding of information, content-specific encoding, interaction tasks and interaction techniques (see Figure 40). The model was adopted and refined by the ISO TC159/SC4/WG9 group for the new standard 'ISO 9241-920 Ergonomics of human-system interaction – Guidance on tactile and haptic interactions' [vEK08]. It was set as a Draft International Standard in September 2007 and it contains the following refined categories: tactile/haptic inputs, outputs, and/or combinations; attributes of haptic and tactile encoding of information; content-specific encoding; design of tactile/haptic objects and space and

	Full 6DOF	Application	Other Tasks Possible	Compatible for Haptics	Taxonomy
Flying Vehicle (2-3DOF device)	yes	non-immersive	no	possible	d-u-r
Flying Vehicle (6DOF device)	yes	immersive /non-imm	no	yes	d-u-r
UniCam	no	non-immersive	no	possible	d-u-r
Camera In Hand	yes	non-immersive	no	yes	d-u-a/r
Treadmills	no	immersive /non-imm	no	yes	d-u-r
Gestures	yes	immersive /non-imm	Sel/Manip	no	d-u-r
Gaze Directed	no	immersive /non-imm	no	no	d-u-r
Eyeball In Hand	yes	immersive /non-imm	no	no	d-u-a
World in Miniature	yes	immersive /non-imm	Sel/Manip	possible	d-u-a
Speed Coupled Flying	no	non-immersive	no	possible	d-u-r/d-o
Scene In Hand	no	immersive /non-imm	no	possible	d-o
Head Tracked Orb Viewing	no	immersive	no	no	d-o
Teleportation	no	immersive /non-imm	no	no	i
Small Scene Manipulation	no	immersive /non-imm	no	no	i

Table 3: Camera metaphors and their support for haptics, after De Boeck et al. [DBRC05]

interaction (more details in 2006 Draft [vECN$^+$06]). In particular, the interaction techniques are classified independently of interaction tasks and relative to the action towards the object of interest (possessing, touching, moving the object, moving relative to the object) plus the additional type of action 'gesturing'.

While previous methods attempt to formulate guidelines for the design process of haptic interactions, Bjelland and Tangeland [BT07] give recommendations for the prototyping of haptic user interfaces as part of a user-centered design process, in particular in the early stages of development and for commercial purposes. Their recommendations include: build on the tradition of user-centered design, prototype from day one, substitute technology, build several different prototypes, develop a vocabulary, and stick with the heuristics.

4.3.3 Visualization

There has been many taxonomies proposed for traditional visualization, categorizing the properties of the data by type [Ber83, CM97] or both by task and type [WL90, Shn96], or categorizing the design models instead of the data [TM04]. All this effort shares the same objective of producing a framework that integrates a 'catalog' of existing visualization techniques and that enables a designer to find the appropriate existing technique or create new ones for a given problem. Although some of these taxonomies focus on visual representations of the data, most of them can be transferred to the haptic domain. For instance, Schneiderman [Shn96]'s information seeking mantra "overview first, zoom and filter, then details on demand" focuses on the user's behaviour and cognitive processes and could thus be applied to any modality.

But, little work has been done to formalize the design of interactions in the context of haptic visualization. Nesbitt [Nes05a] developed a framework of the multi-sensory design space called the MS-Taxonomy, based on different levels of abstraction to allow for comparison between senses. The 'metaphors' are divided into three classes: 'Spatial Metaphors' (perception of properties that depend on space), 'Direct Metaphors' (perception of properties of the sensory modality) and 'Temporal Metaphors' (perception of properties that change over time). This taxonomy, which provides design possibilities, is used in the MS-Process (see Figure 42), a methodology to help the design and implementation of an information visualisation. The main steps of the MS-Process include 'Task analysis', 'Data characterisation', 'Display mapping', 'Prototyping' and 'Evaluation' and

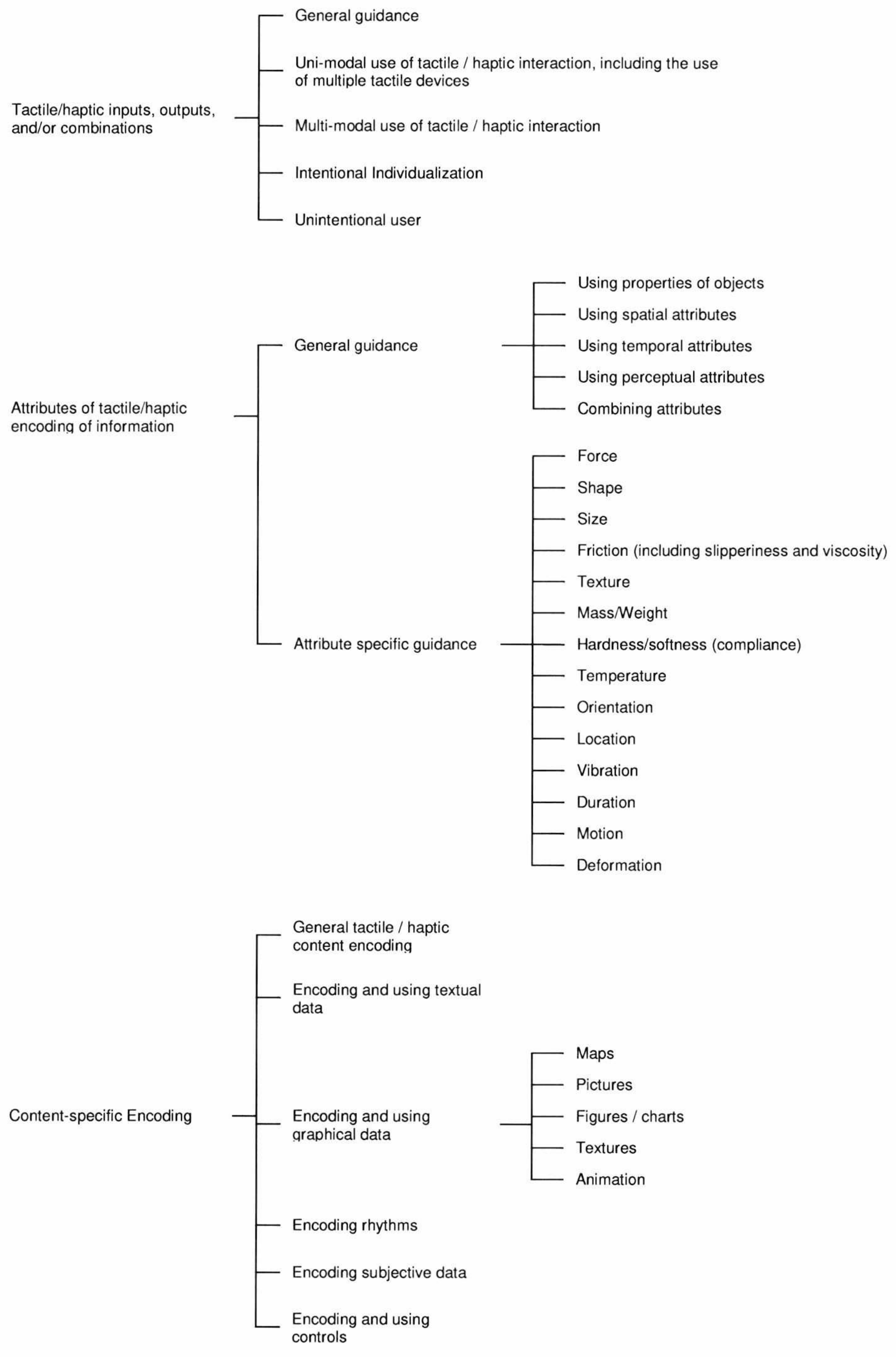

Tactile/haptic inputs, outputs, and/or combinations
General guidance
Uni-modal use of tactile / haptic interaction, including the use of multiple tactile devices
Multi-modal use of tactile / haptic interaction
Intentional Individualization
Unintentional user
Attributes of tactile/haptic encoding of information
General guidance
Using properties of objects
Using spatial attributes
Using temporal attributes
Using perceptual attributes
Combining attributes
Attribute specific guidance
Force
Shape
Size
Friction (including slipperiness and viscosity)
Texture
Mass/Weight
Hardness/softness (compliance)
Temperature
Orientation
Location
Vibration
Duration
Motion
Deformation
Content-specific Encoding
General tactile / haptic content encoding
Encoding and using textual data
Encoding and using graphical data
Maps
Pictures
Figures / charts
Textures
Animation
Encoding rhythms
Encoding subjective data
Encoding and using controls

Figure 40: Summary of the GOTHI model described by Carter et al. [CFF+05] (part 1).

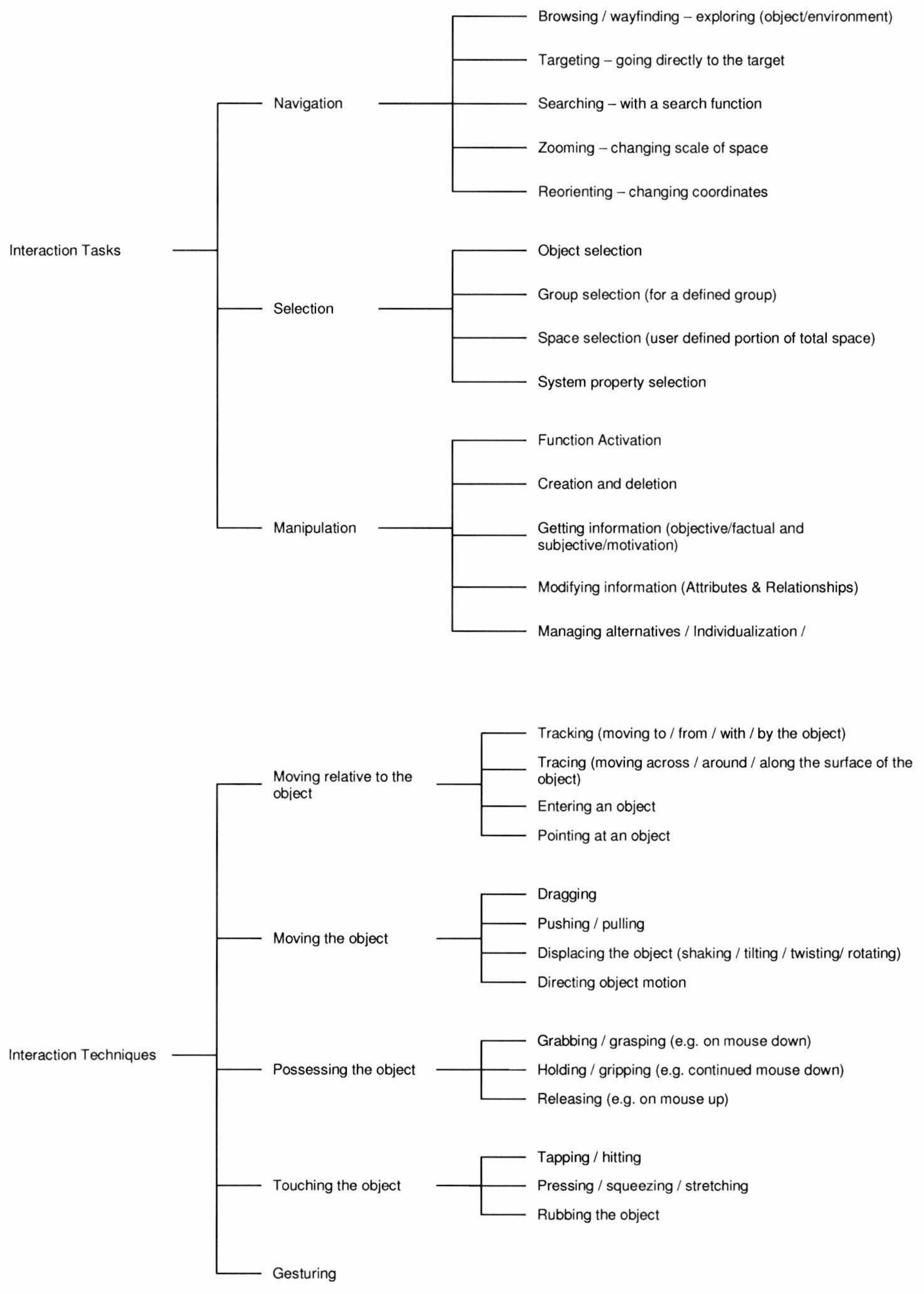

Interaction Tasks
Navigation
Browsing / wayfinding – exploring (object/environment)
Targeting – going directly to the target
Searching – with a search function
Zooming – changing scale of space
Reorienting – changing coordinates
Selection
Object selection
Group selection (for a defined group)
Space selection (user defined portion of total space)
System property selection
Manipulation
Function Activation
Creation and deletion
Getting information (objective/factual and subjective/motivation)
Modifying information (Attributes & Relationships)
Managing alternatives / Individualization /
Interaction Techniques
Moving relative to the object
Tracking (moving to / from / with / by the object)
Tracing (moving across / around / along the surface of the object)
Entering an object
Pointing at an object
Moving the object
Dragging
Pushing / pulling
Displacing the object (shaking / tilting / twisting/ rotating)
Directing object motion
Possessing the object
Grabbing / grasping (e.g. on mouse down)
Holding / gripping (e.g. continued mouse down)
Releasing (e.g. on mouse up)
Touching the object
Tapping / hitting
Pressing / squeezing / stretching
Rubbing the object
Gesturing

Figure 41: Summary of the GOTHI model described by Carter et al. [CFF+05] (part 2).

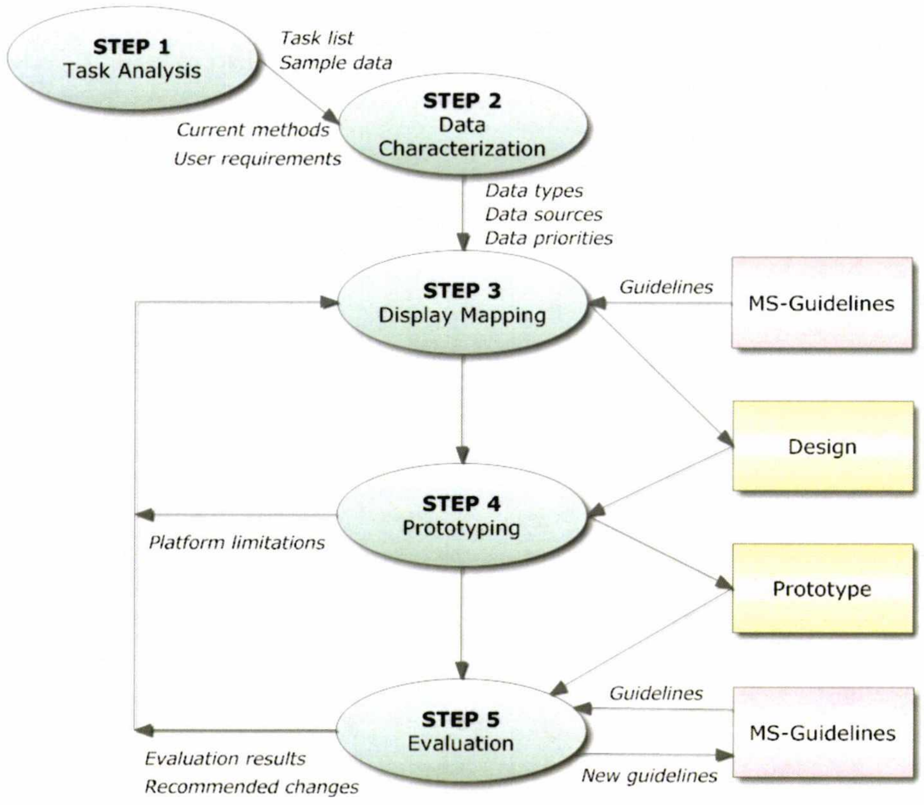


Figure 42: MS-Process after Nesbitt [Nes03] to help in the design and implementation of multi-sensory interfaces for visualization.

are further detailed in [Nes03], Chapter 10. The MS-Process also relies on the MS-Guidelines, which consist of a set of guidelines to support designers of multisensory displays in their design choices. The haptic guidelines [Nes05b] include two parts: general guidelines (issues of perception, information design and multisensory display) and guidelines structured around the MS-Taxonomy (spatial, direct and temporal metaphors). For instance, one guideline explains that "we lose track of spatial location", making "accurate tracking of position in space difficult" haptically or that "Haptics is concerned with movement", suggesting "that haptic movement events may be an appropriate way to display information" [Nes05b]. This combination of taxonomy, guidelines and design process can be a valuable toolset to help the designer in developing haptic interactions.

4.4 Prototyping Languages

This section describes the background research relating to prototyping languages for interactions that enable users to rapidly develop and test several designs. This research is divided into the two subsections: general prototyping and haptic prototyping.

4.4.1 General Prototyping

In the field of Virtual Reality (VR), there are various tools that allow developers to easily create 3D models. Tools like Blender [Ble10], 3ds Max [3ds10] and Rhino [Rhi10] create specific models that can be loaded into virtual environments and navigated by the user. Although these systems allow developers to quickly build virtual environments, it is difficult for developers to experiment and develop novel interaction methodologies and utilize new interaction devices. This is why many researchers have investigated the rapid prototyping of user interfaces, focusing on interactions or the devices adaptability.

In fact, several languages have been created for virtual worlds, which allow the developer to foster the development of new interaction techniques and build a library of reusable 3D interactions. However, most do not integrate the haptic modality. For example, the XML language InTml, by Figueroa et al. [FGH02], aims to describe interactions in a standard way by being high-level, toolkit-independent, reusable and extensible. It is based on a dataflow architecture and has virtual reality objects, devices and interactions techniques as components. It was later extended by Mejia et al. [MFH05] to include a graphical prototyping environment aimed at designers, who only need to connect the graphical components to each other through input and output ports. An observational study conducted with 16 participants showed "an increase in the productivity in the development of VR applications" [MFH05]. However, InTml does not deal with the properties of the objects, and in particular does not address haptic properties, which are required for the design of haptic interactions. Similarly, Ray and Bowman [RB07] are developing the 'Interaction Framework For Innovation' (IFFI), that assists in the development and reuse of 3D interaction techniques across applications. They, however, put an emphasis on considering the non-portable components of a technique and on allowing control over the dependency of the interaction technique to the VR toolkit. With the same concerns in mind, Csisinko and Kaufman [CK07] present a standard implementation of interaction techniques directly in tracking

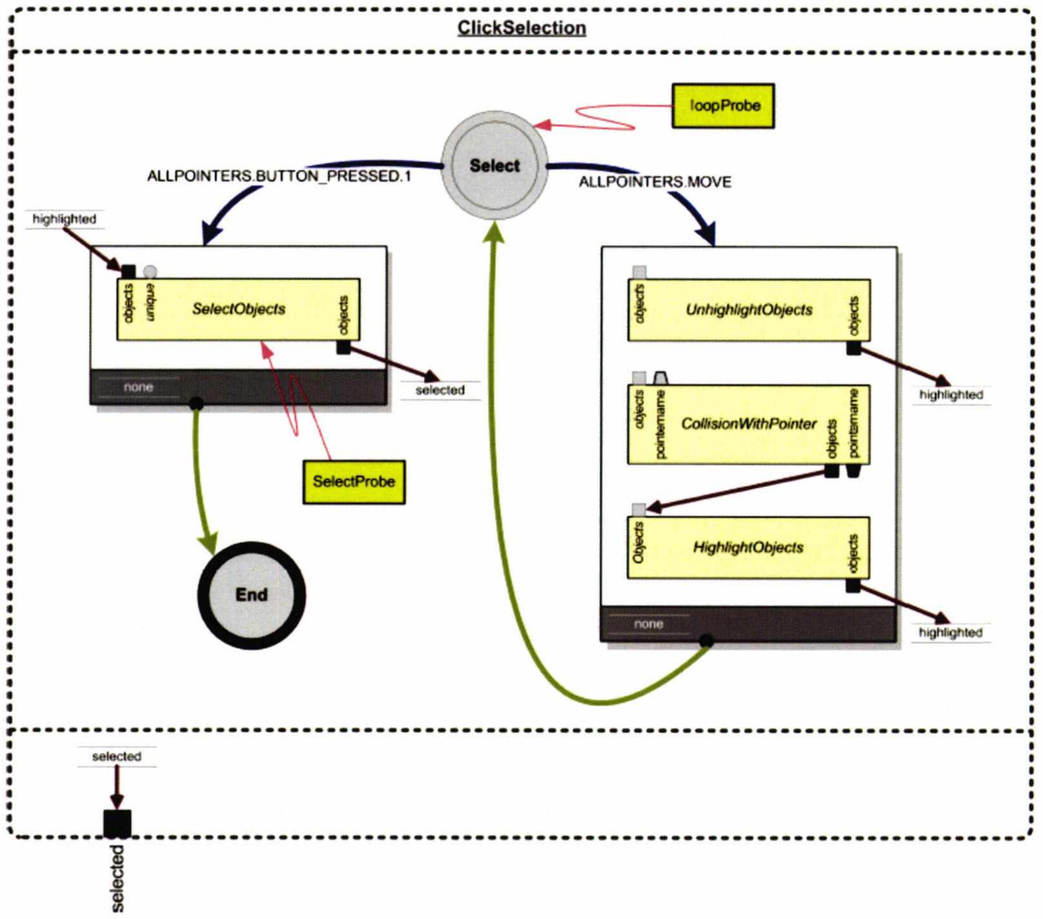


Figure 43: Example of NiMMiT diagram for a highlight by touching and select by clicking interaction technique, from [DBVRC07].

middleware, using Python linked to OpenTracker, which is "not depending on other third-party toolkits"; and considers the level of support achieved in the middleware, i.e. the level of independence of the interaction components to the application. Recent work by DeBoeck et al. [DBVRC07] has proposed a high-level graphical notation called NiMMiT to specify multimodal interaction techniques (see Figure 43); thus including haptics. This software not only allows design to be performed at a high-level, but also allows its automatic execution. As opposed to previous work, NiMMiT is both state- and data-driven. However, output modalities, such as haptics, can only be "added in a diagram through the addition of a custom task which is scripted or coded"([SJGL08], p86).

Lewis et al. [LKL91] specified the behaviour of a virtual world "by a dialogue composed of a number of modular subdialogues or rule sets" in order to ease development and allow for dynamic flexibility and reusability of substitutable devices. An alternative approach by Jacob et al. [JDM99] presents a model for describing "non-WIMP" interactions by combining continuous relationships with a dataflow component (i.e. mouse gesture) and discrete interactions with an event-based component (i.e. pressing the keyboard). Their visual language, combining a dataflow graph editor and a state diagram editor, is however, very low-level and detailed, thus tending to result in screen clutter. Hendricks et al. [HMB03] argued that implementing a single authoring tool that would enable the creation of any virtual reality (VR) application is impossible; therefore they propose a "meta-authoring tool that would rapidly generate a VR authoring system for a particular VR solution". Their tool integrates smooth user migration support (i.e. supporting both novice and experienced users), support for VR applications of any complexity, much shortened development time and configurability. These features are mostly achieved through the separation between conditions (i.e. events) and actions specified in pairs for a specific context, using GUI-widgets or scripting languages, such as Python or Tcl, to program the interactions and behaviour of the world [HMB03].

The re-mappability of devices to interaction techniques or tasks has been the focus of many researchers. Ballagas et al. [BRSB03] developed the iStuff toolkit, which "support[s] user interface prototyping in ubiquitous computing environments" with dynamic reconfiguration at run-time and aiming at integrating multiple devices and multiple simultaneous users in the domain of interactive rooms. The iStuff interfaces several lightweight wireless devices, including buttons, sliders, and wands. However as the focus is on lightweight wireless devices, iStuff does not readily integrate haptic devices, such as force-feedback devices. Also, from the presented examples, the interactions appear restricted to mapping a device to control a widget, such as a virtual paddle, or a functionality such as controlling a high-frequency filtering mechanisms for music, and not more complex interactions techniques. The primary goal is the ability to (re)configure the devices to a simple task (e.g. switching on the lights in the room).

Similar to Ballagas et al., Dragicevic and Fekete [DF04] implemented the Input Configurator (ICon) toolkit, which aims to allow 'Post-WIMP' interaction techniques by providing high-level 'input adaptability'. Input adaptability is defined as "the ability of an application to exploit alternative sets of input devices effectively and offer users a way of adapting input interaction to suit their needs". The toolkit defines interactions in terms of devices with the flow of the input/output values. Similar to Ballagas et al., the interaction techniques seem to follow a direct or simple mapping between the devices and other devices or functions (scrolling, voice commands, etc.). It is unclear how complex interaction techniques, such as those involving haptics, would be easily integrated. Moreover, this tool seems rather unintuitive for people with no or little programming knowledge.

In fact, several researchers are focusing on post-WIMP interfaces, and creating various tools and toolkits that enable users to assemble and link components together. Huot et al. [HDD$^+$04] developed the MaggLite toolkit for fast and interactive design of post-WIMP user interfaces. It uses a 'mixed-graphs' approach which separates the definition of the presentation from interactions, using a scene-graph and an interaction-graph that are dynamically combined at run-time. The interactions are specified using the ICon notation while the presentation is performed using drawing tools. Examples of more advanced interaction techniques are described; however the toolkit is restricted to 2D presentations.

Navarre et al. [NPDB06] explain that with the ICon notation alone it is "difficult to deal with the representation of the set of states the system can be in and how events produced by the user through the use of input devices make the states evolves". Hence, they integrated the ICoM model (and ICon environment) with the ICO formal description technique (and the PetShop environment), which through Petri-nets describes the system's states and their changes. Similarly, Appert et al. [AHDBL09] created the FlowStates toolkit, which combines ICon with the Java Swing extension SwingStates, to allow easy prototyping of complex interaction techniques. ICon is also used for device configuration while programming using SwingStates enables the addition of machine states. However, these techniques involve some programming, making them inaccessible to non-programmers.

Bouchet et al. [BNB04] describe a component-based approach, called 'ICARE', for specifying and developing interfaces combining active (user commands) and passive modalities (context information) for mobile and/or pervasive systems. This approach is aimed at designers rather than developers and automatically generates code on creation of an ICARE diagram. The ICARE platform includes the following components: devices (abstraction of physical devices data); interaction language; and composition components (complementarity, redundancy, equivalence and complementarity/equivalence). They illustrate their approach with three application examples. Figure 44 summarizes the ICARE diagram for the augmented reality Memo application on a PDA.

Influenced by ICon and their previous attempt ICARE, Serrano et al. [SNL$^+$08]

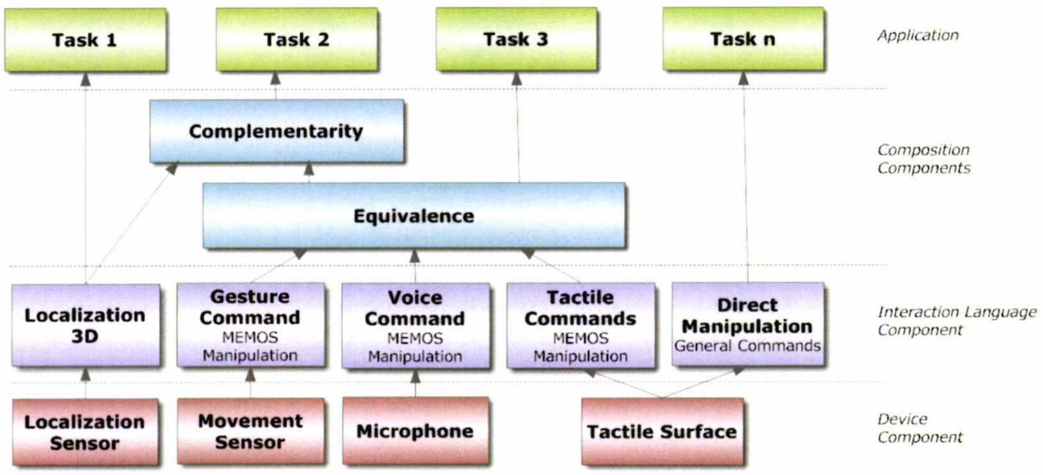


Figure 44: ICARE diagram for a Memo-PDA application. Task 1 displays the memos that belong to a circle which has a center that is the user position. Task 2 corresponds to the addition of a new memo to the real world, while Task 3 deals with the consultation, editing and deletion of memos. Task n corresponds to other tasks achievable with the tactile screen. Task 2 is achieved with the two events providing the user localization and the activation command by either of the three input modalities, using the complementarity composition. The equivalence composition means that any of the modality can be used to achieve the task, e.g. deleting a memo can be done with a vocal command, a gesture or a tactile command. This diagram was translated and adapted from [BNB04].

underline that "the main limitation of ICon is that the level of abstraction of its components is too low and assemblies of components become too complex" while for ICARE "only a limited set of [the] components were really generic". Therefore, they adopted a "more realistic mixed-approach" combining generic components and components tailored to an application [SNL$^+$08]. They describe a characterization space of software components along three dimensions [SJN08]: component genericity (reusable, device-dependent or application-dependent components), data-flow from devices to an application (device, transformation, composition from ICARE and task) and the component approach (software component, including technical details for programmers and interaction entity, i.e. general characteristics for the designers). Based on these characteristics, they developed the OpenInterface (OI) framework that "focuses on providing an interaction-independent, device and technology independent flexible solution for the fast prototyping of multimodal applications through the facilitation and reuse of existing software and technologies" [LAAVM09]. The OI framework is based on a kernel, which is component-based and integrates components that can be written in Java, C/C++, Matlab, Python and .NET. XML-based descriptions are used to interface these components, which can be assembled to create a 'pipeline' for the definition of a multimodal interaction using the graphical environment OIDE [SNL$^+$08]. As OIDE has several shortcomings, including inflexible design and not enough influence of non-developers, Lawson et al. [LAAVM09] developed the SKEMMI graphical interface, with support for components development, support for multi-level design involving users with different backgrounds, support for reusability, support for documentation, and runtime and debug functionalities. Informal evaluations demonstrated that the framework could be successfully used to prototype various applications. However, it is unclear how haptics can be easily integrated into this framework.

Finally, specifically in the field of tabletop and multi-touch surfaces, various researchers are investigating high-level visual languages to control the functionality and to integrate image processing with touch-based methods. For instance, König et al. [KRR09] describe the interaction library Squidy, with the goal of unifying multiple toolkits and various tangible objects (such as pens or physical tokens) for the design of interactions for natural user interfaces such as tabletop displays. Squidy includes support for visual semantic zooming, allows the tuning of the parameters or the code on the fly, as well as the display of information about the nodes.

Facilitating the design of the novel types of interfaces, often referred to as non-WIMP, which integrate novel and more complex interaction techniques and modalities, is increasingly becoming a topic of interest, and particularly the investigation of high-level languages. For instance, several workshops have been organized, including the CHI workshop on User Interface Description Languages (UIDL) [SJGL08] which collates many more interesting references, and a special issue of the ACM Transactions on Computer-Human Interaction [SJGL10] with the same editors will soon be published.

4.4.2 Haptic Prototyping

Although most of the prototyping frameworks described above allow the integration of a wide range of devices, it is unclear whether haptic and, in particular, force-feedback devices would be easily supported. Therefore, solutions specific to haptic prototyping are also discussed in this chapter.

In the tactile domain, several tools have been developed to facilitate the prototyping of vibrotactile icons, as described in Section 2.4.1, in particular, Lee et al. [LRC09] who use a musical metaphor so the user does not have to deal with the low-level specifications of the vibrotactile signals.

Rapid prototyping of haptic worlds has also been the focus of various researchers. Rossi et al. [RTW05] designed a tool for the prototyping of haptic and telehaptic applications, built on top of the Matlab/Simulink platform. Their sample example exhibited the use of a graphical VRML authoring tool to model the 3D graphical environment and a block based diagram approach to add haptic effects and model time transformations. Forrest and Wall [FW06] developed a haptic prototyping tool that enables non-programmers to build a haptic 3D model with the haptic device. Complex models can be created by combining primitives, whose size and rotation can be changed through the use of edit points, using a device from the PHANTOM family. An evaluation, with seven participants, including novices and experts in using the PHANTOM device, showed that the participants could construct 3D models within the time allotted. It also highlighted some areas of improvements, such as depth perception, which will be dealt with in the future work. Kurmos et al. [KJR09] uses the Scene Authoring Interface (SAI) to integrate haptics into an X3D authored virtual world.

These techniques mostly deal with the haptic modelling of an environment, and not with the behaviour or interactions in this environment. This is why the HAML framework [EAS06] aims to provide a fast prototyping environment that hides the complexity of haptic programming. HAML describes all the components involved in an application (application general information, haptic device and its capabilities and limitations, the haptic and visual rendering, the haptic API, quality of experience and haptic data) in an XML-based language that is used to dynamically generate the application from the user's requirements. After defining the structure and description schemes for the language, Eid et al. [EAAE08] developed the HAML-based Authoring Tool (HAMLAT) to allow non-programmers to create visual-haptic worlds, by extending Blender, a 3D graphic modelling software, to support haptics. However, the preliminary work restricts itself to static scenes with no dynamic behaviour.

De Felice et al. [DFAD09] present an authoring tool to design the haptic/acoustic user interface of a VE explored by visually impaired people. The tool is based on the decoupling of representation and interactions and targets the design of the latter. The tool uses a model where the scene is a series of scenarios, containing active objects (scene objects with acoustic, haptic properties or a combination of both) and guided paths. The visual editor allows the specification of object behaviour by assigning properties and events (translation/rotation). They also discuss the interaction metaphors transferrable to the non-visual domain. Although this approach seems very interesting, it lacks details about the actual tool and the type of prototyping it supports other than attributing properties to objects.

As for commercial tools, the Immersion Studio [Imm10] allows rapidly adjusting tactile and force feedback effect parameters for all the consumer TouchSense gaming products. The LabVIEW visual programming tool has been used with the National Instruments PXI system in an undergraduate mechanical engineering course in system dynamics as a data acquisition and processing platform for haptic paddles hardware [BO06] and for both teleoperation research and haptic device development at Georgia Tech University [Boo08]. LabVIEW has been used extensively in designing control and analysis solutions in the area of engineering technology and education. However, as it is aimed at engineers and scientists, the graphical notation is quite low-level and focuses primarily on signal acquisition, processing and analysis rather than designing computer interactions, in particular 3D haptic virtual interactions.

4.5 Summary

This chapter presented the different methodologies proposed to facilitate the design and development of interaction techniques, in particular in regard to the haptic domain. These methodologies range from taxonomies and frameworks, that provide classifications, guidelines and processes for the design and testing of interaction techniques to prototyping languages that help the user author virtual (and haptic) worlds, interaction techniques and vibrotactile icons.

This review also highlights that taxonomies and frameworks have been well investigated, especially in the virtual reality domain. Although compiling these taxonomies and methodologies that apply to the haptic domain with guidelines and frameworks specific to the haptic visualization area would be beneficial, this is not the focus of this thesis. The new standard 'ISO 9241-920 Ergonomics of human-system interaction – Guidance on tactile and haptic interactions' [vEK08] covers the issues related to tactile and haptic interactions and therefore will include many guidelines applicable to haptic visualization. Developing a framework for the development of haptic interactions for visualization, i.e. drawing inspiration both from Bowman and Hodges [BH99] and from Nesbitt [Nes03], is perhaps a more interesting challenge that has not been much investigated and will be left for future work.

Moreover, the review exhibited that few prototyping languages have been proposed in the haptic area that clearly support force-feedback devices. Considerable work has been established for virtual reality applications with a focus on device configurability, however the examples provided do not clearly account for force feedback devices. In the haptic area, some work has been done to facilitate the design of vibrotactile icons and for the authoring of haptic representations. Nonetheless, no work has been found on the prototyping of haptic behaviour for force feedback devices. The HAML framework [EAS06] plans to integrate the prototyping of haptic behaviour as part of the prototyping environment, but their work is still in its early stages.

Therefore, the rest of thesis focuses on the development of a haptic interaction prototyping tool, for force-feedback interactions. Additionally, this tool will be designed so that it is accessible to programmers that are new to the haptic API (as advanced programmers can develop and test interactions more quickly) and people with no or little programming knowledge. This is motivated by the fact that most of the prototyping languages presented were aimed at developers, and when access was envisaged for designers, programmers were still needed to implement the components necessary to view the results. This ensures greater flexibility, but does not enable designers to rapidly view and alter the results of their design.

4.6 Prototyping Language Model

This review has emphasized several concepts which will be reused in the design of the prototyping tool.

First, as the tool concentrates on prototyping interaction techniques, the tool will use the separation of the scene description from the definition of interactions, which will then be combined at run-time. This is similar to the approach used by MaggLite [HDD$^+$04] or De Felice et al. [DFAD09]. As explained at the end of Chapter 2, representation and interactions are two of the main components of an application design. They influence each other, but they can be treated separately during development. This is why static properties (geometric and haptics) can be specified through the scene description, while dynamic properties related to active exploration can be defined with the interactions prototyping. The tool will therefore focus on the behaviour specification, but will allow for 'communication' with the scene, created externally. In an initial stage, the scene will be described as a scene-graph, nonetheless, in a later stage, the tool could integrate a module for prototyping the scene, such as Eid et al. [EAAE08].

Secondly, the prototyping language will be based on a dataflow model which separates the conditions/events from the actions to ensure greater flexibility in the specification and combination of behaviours. This is similar to the approach of Hendricks et al. [HMB03] and also to some extent to the ICARE composition components (complementarity, redundancy, and equivalence) which provide and/or relationships. This should allow for a prototyping language closer to natural language, e.g. "wait for X to happen and then if the right conditions are met, do Y", which would be accessible to non-programmers. The actions will correspond to tasks for navigation, selection and manipulation as well as haptic primitives, while the conditions/events will allow the control of the flow of these actions.

Lastly, the tool aims at being high-level to be accessible to non-programmers. However, the high-level stage of the model does not provide access to lower-level components and greater flexibility. This can be achieved by automatically generating code in a scripting language, such as Python. Therefore, beginners to the haptic API can use the tool as a first stage to learning how to program with it and extending the functionalities of the interactions, while designers can provide this skeleton code to the programmers with a clearer idea of the behaviour they want them to achieve. Hendricks et al. [HMB03] described this ability to support novice users during the learning process and allow them to evolve from a beginner's stage to a more experienced stage as a "smooth integration". Likewise, they provide both graphical components (i.e. GUI-widgets) and scripting language functionalities.

The integration of these concepts for the development of the prototyping tool will be further discussed in the next chapter.

Chapter 5

HITPROTO: Haptic Interaction Techniques Prototyping Tool

Chapters 1 and 2 showed that haptics is a growing research area and that the range of available haptic devices is increasing, from custom devices developed by research laboratories to commercial devices. However, developing haptic applications and interactions is still difficult and time-consuming (as explained at the end of Chapter 3), and although various APIs are available (which provide a generic interface to multiple devices), they still require the user to have good programming skills, a good understanding of haptic interactions and technical knowledge of the devices to be used.

Consequently, there is a need for prototyping tools that can be used to quickly implement and test haptic interactions. This is a natural continuation, driven by the rapid expansion of haptics. As discussed in Chapter 4, the demand for such toolkits has long been identified and satisfied in related research areas, such as virtual reality, where industry has access to CAD software and, more generally, there are user-oriented authoring tools to create virtual worlds. The ability of designers to quickly design and test interaction metaphors enables the creation of an enriched experience for users of the virtual world.

In the area of haptic data visualization, also referred to as 'haptification', the aim is to provide an understanding of the underlying data through effective data mappings and user interactions using the sense of touch. Rather than merely attempting to render the world, haptic feedback in visualization is used to convey the information about the data being presented through well designed and efficient metaphors. It is useful for visually impaired people or in situations where the visual (or audio) sense is overloaded with information (as described by the examples in Chapter 2 or in [PR09]). However, the use of haptic interactions for visualization is not widespread [PR09, RP07]. Accordingly, a system that would allow the rapid development of haptic interactions, especially in the context of data visualization, should bridge this gap and encourage the development and exploration of new haptic interactions, as well as permitting a wider audience to explore the possibilities of haptic visualization.

This chapter presents HITPROTO, a visual prototyping tool for haptic interactions. The aim of the prototype is to allow developers with no or little programming skills, such as blind students' teachers or designers, to explore interactions and more generally to stimulate the development of interactions to access the data haptically. In Section 5.1, the motivation for developing HITPROTO is explained. Sections 5.2 and 5.3 present the rejected ideas for design and the final design for HITPROTO. Section 5.4 explains how to operate the tool while more details about the implementation are given in Section 5.5. Examples of scenarios produced with HITPROTO are illustrated in Section 5.6.

5.1 Motivation

HITPROTO, or Haptic Interaction Techniques PROtotyping TOol, allows users to perform rapid prototyping of haptic interactions, with an emphasis on data visualization. As highlighted in Section 4.4, there are not many prototyping tools available for developing and testing haptic interactions. The few that could potentially integrate haptics into their framework often describe the blocks using input/output flow which can be unintuitive to program complex interactions. Moreover, many of these tools are not aimed at non-expert users as they require some programming. In contrast, HITPROTO hides the technical complexities and provides an interface that is closer to a natural language (e.g., "Wait for a button press, then add and start guidance"). The main hypothesis is that in doing so, prototyping haptic interactions will become accessible to people with little or no programming knowledge and it will be faster than learning the API and languages to program the device's behaviour for designers and developers.

5.2 Rejected Design Ideas

Initially, after investigating XML-based approaches used in the Virtual Reality area (see Section 4.4.1), using an XML-based language was envisaged. Similar to

IFFI or InTml, the language was defined through input and output variables for a given interaction technique. The user would use predefined interaction techniques from a library and adapt them to their application by defining and linking the suitable inputs and outputs. However, as the main design goal was to let the users prototype interaction techniques themselves to suit their needs rather than providing a predefined interaction technique, and given the increasing complexity of providing lower-level functionalities through XML, this approach was dropped in its early development stage. XML can be useful in many cases, but as soon as the file gets lengthy, it is less intuitive to read and modify, especially for non-technical people.

Therefore, a graphical approach was considered. For instance, the graphical notation NiMMiT [DBVRC07] exhibits an intuitive compromise where the interaction could be prototyped using a diagram-based approach, in turn converted to XML, which could also directly be altered. Subsequently, graphical approaches were investigated. As the targeted user did not require any programming skills to prototype interactions, a high-level block-diagram approach was chosen as being the most intuitive and is described in more details in the following section.

5.3 Tool Design

HITPROTO uses the H3DAPI [H3D10]. This API was chosen as it is a high-level open-source haptic API that interfaces several haptic devices and thus several lower-level APIs. Haptic applications can be programmed using X3D[1] with Python or using C++. However, using an API, even as high-level as H3DAPI, still requires programming skills and a relatively long learning period before being able to program interactions, and therefore restricts itself to developers. HITPROTO aims to significantly reduce the difficulty of designing haptic interactions.

The design of the tool was inspired by various visual programming environments, and in particular the Lego Mindstorms NXT software based on LabVIEW. Mindstorms NXT is a visual programming environment that allows users to program the behaviour of a Lego robot using a dataflow language. Similarly, HITPROTO uses tunable visual blocks to program haptic interactions. These blocks fall into two main categories: action blocks (such as addition, removal,

[1]X3D [Web10] is an ISO open standard scene-graph design and the successor to the VRML standard. It is used and extended within H3DAPI to specify the 3D scene-graph of the world and particularly the geometric and basic haptic properties.

creation and modification of haptic and guidance effects) and flow blocks (waiting for an action, looping and conditionals) that allow controlling the flow of the data by listening to events (e.g. "Wait for the device's button to be pressed") or by testing conditions ("If the key pressed is equal to 1"). These blocks are abstractions of elements (or combinations of them) available in the H3DAPI. Setting the parameters of these blocks and linking them together enable the production of an executable scenario. The scenario is written in Python code which can be directly executed or that could further be used as a code skeleton that could be later extended, or to learn the H3DAPI.

As summarized in Section 4.6, HITPROTO is based on the separation of the scene description and the definition of interactions (such as MaggLite [HDD$^+$04]) and focuses on the specification of the latter using the visual blocks. These blocks separate the conditions/events (flow blocks) from the actions (action blocks) to ensure greater flexibility in the combination of behaviours and to allow for the definition of interactions in terms closer to natural language, in order to be accessible to non-programmers. These non-programmers can thus prototype simple interactions and use the generated Python code to progressively learn how to use the API or as a skeleton code to extend the interaction.

The available blocks are summarized below, along with their functionality. The icons representing the block shapes were chosen to reflect the name of the block name. For instance, the *Switch* block is represented by a physical switch; the *Guidance_Add* block is pictured by a map and a compass while the *Wait For* and *Everytime* blocks are represented by a timer and a clock with a looping arrow respectively, as can be seen below.

Action Blocks

Stop: compulsory block that delimitates the end of the 'interaction scenario' or program. There are no parameters. Several lines can be connected to it, but none can depart from it.

Guidance_Add: creates (and can add to the scene) a guidance instance, that needs to be named. A guidance instance includes a spring to attach the device, an anchor to visualize the spring and parameters such as path and speed/duration.

Guidance_Control: enables the control of a guidance instance using its name, by starting, pausing, resuming (after a set time) or stopping it. When the guidance is paused, the parameters of the spring can be changed, to allow for wider movement for instance.

Haptic Effect: creates (and can add to the scene) a chosen haptic effect, which needs to be named. The available haptic effects are: SpringEffect, Magnetic Line(s) and PositionFunctionEffect. The PositionFunctionEffect enables the specification of the components of the force on each axis or according to a model.

Add_Modify: allows the addition of a previously removed object, the addition of a created but not yet added instance or the modification of the parameters of an existing instance using its name.

Trash: enables the removal of an object using its name. The removal does not delete the object itself, which can be re-added using the Add_Modify block.

Highlight: enables the haptic "highlighting" of a named object from the scene by adding a spring to the object, making it magnetic or surrounding it with a magnetic bounding box.

Unhighlight: enables the removal of a highlighting effect. The option 'any effect' can be chosen in cases where multiple effects have been used.

Select: enables the tracking of the selected object by putting the name of the object into memory by storing it in the selected name.

Flow Blocks

Wait For: enables the interruption of a sequence of actions until a chosen event happens. The events can be: the device button being pressed/released, a keyboard key being pressed/released, a mouse button being pressed/released, an elapsed time or the activation of a spring (from a guidance or a haptic effect instance).

Everytime and **Everytime_end:** enables the execution of a set of actions specified within the two blocks every time a chosen event occurs. The events can be: the device button pressed/released, the keyboard key(s) pressed/released, a mouse button pressed/released, an elapsed time, the haptic device touching an object ("Pointer Collision") and monitoring the state (position/time) of a guided movement ("Movement"). The "Movement" option should be used in conjunction with the Switch block with the corresponding option. If desired, it is possible to leave the loop by setting a leaving condition such as a number of occurrences or time.

Switch and **Switch_end:** allows testing to see if a condition is satisfied or not before executing a set of actions contained between the two blocks. It is used after a Wait For or Everytime block. The Switch block has two (exactly two, even if a sequence is empty) lines departing from it: the upper line represents the "if chosen condition is satisfied" statement while the lower line represents the "if chosen condition is not satisfied statement". There are four main tests: *Keyboard* - the value of the key pressed; *Logic* - value of some of the parameters of the Guidance_Add and SpringEffect from the Haptic Effect blocks; *Movement sensor* - when used in conjunction with the Everytime block to test the value of the current position or elapsed time and *Compare* - to test if some values are equal.

5.4 Using HITPROTO

HITPROTO is divided into four regions (Figure 45): a menu bar (i.e. Save or Open File), a left panel, a canvas (diagram) and a bottom panel.

The **menu bar** allows for the common operations of creating a new file and saving it, or opening an existing file. It also enables the user to edit and run the

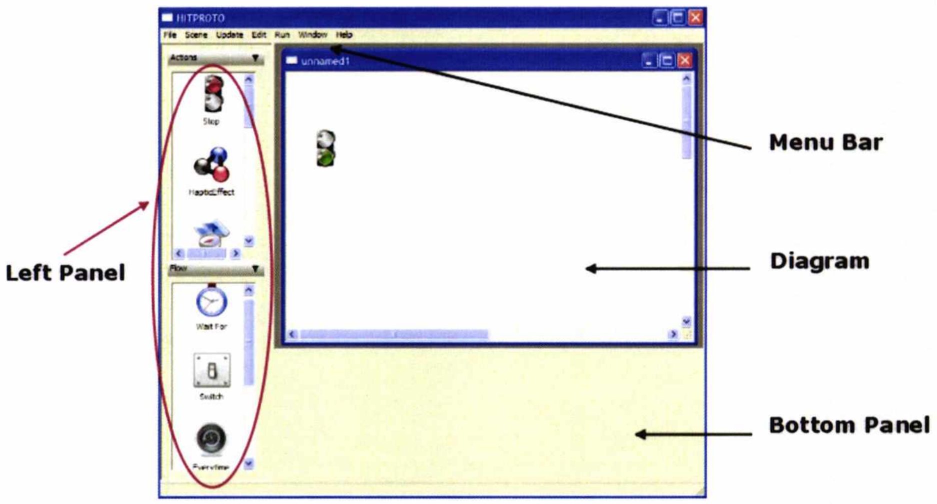


Figure 45: The four main regions of the tool: menu bar, left panel, diagram and bottom panel.

diagram, as well as loading a scene. In particular, for line chart and scatter plot visualization, a module called "VizModule" is available in the scene menu and allows the automatic creation, from a set of points, of an X3D scene, which the user can then load and for instance, create interactions for it such as involving guidance (see Section 5.6) or the addition of haptic effects (see Section 6.1.2 for a scatter plot visualization example). In the case of a line chart scene, the same representation used in Chapter 3 with the engraved modelling (choice of V or U-shape), embossed axis and walls surrounding the chart, is generated. For the scatter plot scene, the data points can be extracted from text files or EXCEL files and the module enables the user to name the different datasets for later reuse in HITPROTO. The scene simply displays the point clouds as a set of points with a color associated to each dataset, no 'haptic representation' is yet available.

The **left panel** contains the block shapes, divided into two expandable list menus, namely 'Actions' and 'Flow' blocks. The list menus contain the icons of the block shapes as well as their names.

To populate the **canvas**, the user can drag'n'drop the chosen block shapes onto it. Apart from the *Start* and *Stop* shape, each of these blocks has a set of parameters which the user can tune to suit their needs (e.g., when adding a spring effect, the developer can tune the spring constant, the spring position and the spring force range). These parameters are displayed in the **bottom panel** when the block is selected (the parameters for each block are available in Appendix B.4).

To test the constructed interaction diagram, the user needs to appropriately link the shapes from *Start* to *Stop* and then run the diagram.

5.5 Implementation

There are two main parts to the implementation: the HITPROTO GUI and the Python code generation (see Figure 46).

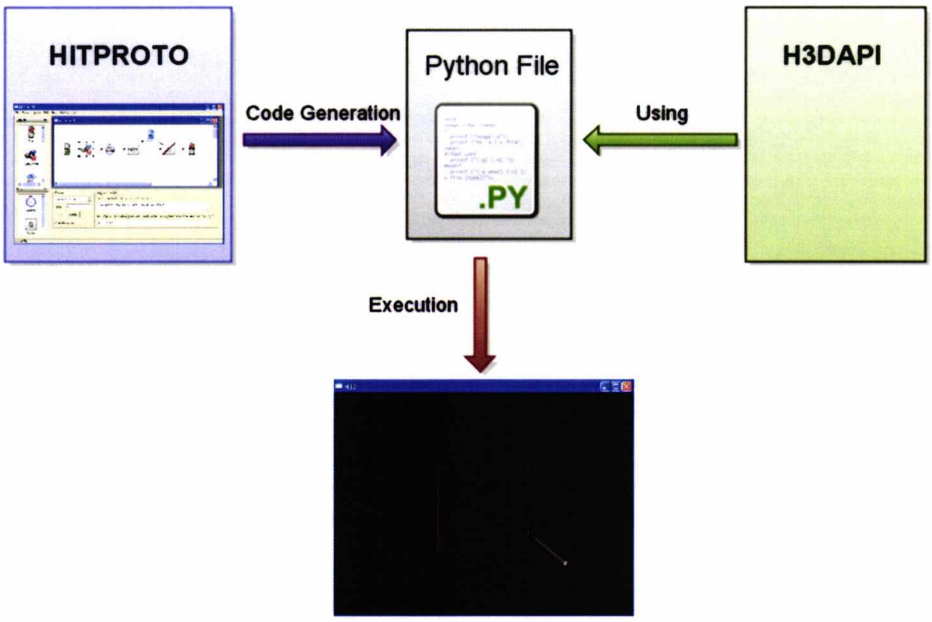


Figure 46: Diagram showing the different components involved in the creation and execution of an interaction scenario.

HITPROTO has been implemented in C++ with WxWidgets for the graphical user interface. With the exception of when a scene is loaded and of operations related to scenes (such as checking whether the pointer touches an object), the visual blocks are implemented independently of the H3DAPI, which is only used to generate code.

The blocks and their parameters are based on elements and functions available in the H3DAPI, but are not a direct translation from the elements of the API. For instance, the *HapticEffect* block encompasses the different effects provided by the API, including a constant force, magnetic lines or a spring effect. The parameters for these effects are based on the input parameters used to define them in the API. As an example, a spring effect is defined by its position, the start distance of the effect (the distance defining the radius where the effect starts), the escape distance (once the device is attached, the distance defining the radius where the effect stops) and the spring constant. The *Guidance_Add* and *Guidance_Control* blocks on the other hand are a combination of several elements from the API, including a geometric representation, a spring force, and a time sensor and a position interpolator for the movement. Other blocks such as *Trash* or *Add_Modify* are more general actions that simply remove or add/modify specified objects. The blocks correspond to tasks (such as highlight, guide or remove object) and haptic effects.

In the H3DAPI, the nodes corresponding to the blocks also have outputs. For instance, the output values for a spring effect include whether the spring is active (whether a device is attached to it) or the spring force generated. During the design and implementation of the dataflow block-based language, the issue of visually providing inputs/outputs for each block to allow for linking between outputs and inputs of two consecutive blocks arose, as well as the use of variables to store and act on these outputs. As the tool was primarily aimed at users with no technical skills, keeping the language as simple and intuitive as possible was the priority.

Therefore, output values were integrated directly into the flow blocks using block parameters. There are usually two cases: *outputs of objects*, such as the active state of a spring, are used directly in the flow blocks as a parameter, i.e. '*Wait For* the spring to be active'; and *outputs related to events* are separated from the *Wait For* or *Everytime* blocks which listen to these events to allow for more flexible testing, for example testing which keyboard key was pressed is done with the *Switch* block. Therefore in the latter case, the *Switch* block has to be used in conjunction with the flow block listening to the corresponding tested event, where the connection of the two blocks creates an implicit linking between the output of one block to the input of the other. This solution was judged as being the simplest and most adequate for non-technical users in order to avoid technical and abstract concepts and also because, when programming, the output values are usually used to control the behaviour through events or tested conditions.

Variables were also avoided as much as possible as they are quite an abstract concept for non-technical users. In a few cases, these were introduced as global 'containers' that would store objects. For instance, in the case of object selection or highlighting, providing access to the actual scene shape selected or highlighted was necessary. Consequently, the 'containers' or variables 'selected', 'highlighted' and 'touched' (the last object touched by the device) were used and also directly

```
<Start>
  <line id=''1''>
    <Guidance_Add position=''120, 100'' name=''MR'' addnow=''1''>
      <General path=''-20, -10, 0; -10, -30, 0; 20, 0, 0; 30, 20, 0;
          40, -10, 0'' speed=''30''/>
      <Spring k=''100'' startDist=''10'' escDist=''100''/>
      <Shape vis=''Yes'' type=''Box'' color=''rgb(104, 170, 85)''
          size=''20 20 20'' />
    </Guidance_Add>
    <WaitFor position=''195, 100'' selection=''4'' condition=''0''
        spring=''MR''/>
    <Guidance_Control position=''270, 100'' instance=''MR''>
      <Start checked=''0'' />
    </Guidance_Control>
    <Stop position=''345, 100''/>
  </line>
</Start>
```

Figure 47: Example of a saved XML file for a guidance interaction that starts once the spring of the guidance object 'MR' gets active.

integrated within the suitable blocks (*Switch*, *Select*, *Highlight*, *Unhighlight* and *Trash*). Output values and variables could be integrated in future work to provide more lower level access to the blocks and the interaction definition for advanced users, in a similar way to the Lego Mindstorms NXT software.

After the users link these blocks in an interaction diagram and execute it, each block is parsed and the corresponding Python code is generated and executed with the H3DAPI (see Figure 46). Python provides an easy way to interface the H3DAPI. Therefore, generating Python code was chosen, instead of directly instantiating the different nodes in C++, so that the code could be used as a start skeleton to extend the developed interactions or to more quickly learn how to program with the API for developers new to H3D.

In terms of the actual C++ classes, there are four main set of classes. The first deals with the overall frame, associated subframes, and associated events such as opening and closing the frames, or for mouse events. The second deals with the blocks, their general actions (dragging them, deleting them, displaying the parameters) and their associated parameters, displayed in the bottom panel. The third one deals with saving and opening linked diagrams, which are saved as XML descriptions that include the order of blocks and the values for their parameters. An example of such a file for a simple guidance interaction that starts as soon as the device gets attached to the guidance object is illustrated in Figure 47.

Finally, the last set of classes deals with the code generation. Blocks are parsed sequentially and the corresponding Python code is written in a string associated to the main flow blocks. These blocks of strings are then all written at once in the executable Python file(s). At first, the code was written directly into the file, but navigating in the file to write the output code at the right position was difficult, therefore strings were chosen instead. Indeed, the code generation is not 'linear' as the flow blocks *Wait For* and *Everytime* correspond in 'H3D-Python' to classes that are listening to events, which in turn need to be initiated from the 'main' body of the file or other classes. For instance, the simple sequence that adds a guidance object and waits for its spring to be active before starting it would include: the calls to create a guidance object with the chosen parameters in the main body of the file, creating the class that listens to the guidance instance's spring events, which needs to be placed above the main body in Python, and within that class, calls to start the guidance when the spring becomes active as well as adding the call to that class in the main body.

Algorithm 1 shows the pseudo-code for the recursive parsing algorithm of the block diagram. The algorithm parses the diagram shape by shape and calls the function corresponding to the shape to write the Python code. The algorithm differentiates the cases of 'Action' shapes and 'Flow' shapes as the flow shapes determine the structure of the Python file, as explained above. The code for the 'Action' shapes can either be located in the 'main body' or within a class listening to events, depending on whether they follow a 'Flow' block shape; while *Wait For* and *Everytime* shapes require their own class as well as adding the call to that class either in the main body or within another class. The *Switch* shape is also a particular case as two lines depart from the shape and connect back to the *Switch_end* block. The algorithm checks whether one condition is empty to ensure that the appropriate sequence of 'if-else' is written, as an empty written 'else' or 'if' would cause a faulty executable. Moreover, to ensure that once *Switch_end* is reached the first time the parsing does not continue, 'end_conditions' are used to stop the parsing at a given recursion. As the *Wait For* block does not have a corresponding end block and therefore no end condition, it is treated separately and the corresponding class is 'closed' at the very end in the Python file.

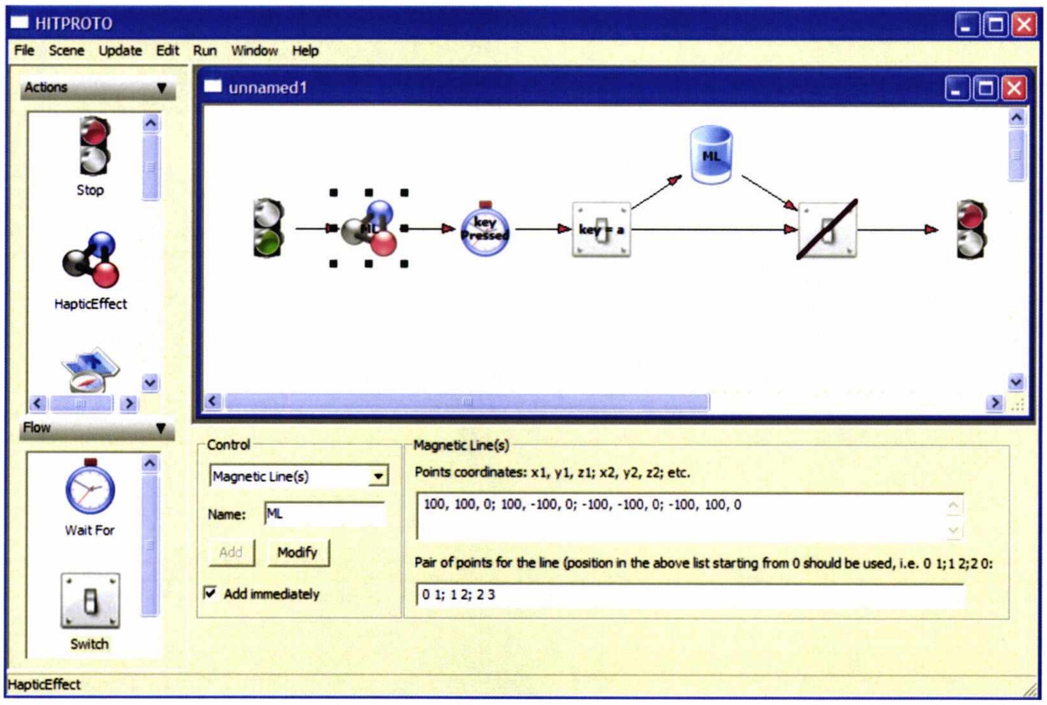
In theory, HITPROTO could be used with any devices supported by the H3DAPI and also multiple devices; however the current system has only been tested with one PHANTOM device so far, the rest being left for future work, once the design of the tool has been validated.

Algorithm 1 Algorithm to parse the block shapes and generate the corresponding Python code.

```
Procedure Parse(shape, end_condition, indent)
 1: line ← first line departing from shape
 2: while line do
 3:   shape_to ← shape the line connects to
 4:   shape_name ← GetName(shape_to)
 5:   if shape_name = end_condition then
 6:     shape_endflow ← shape_to
 7:     if shape_name = "Switch_end" then
 8:       empty ← test if the condition is empty /* case where shape_from_name
          = "Switch" */
 9:       GenerateEndCodeSwitch( shape_to, indentation, empty )
10:     else if shape_name = "Stop" then
11:       GenerateEndCodeWaitFors() /* close 'open' classes for Wait For */
12:       reset the variables for the next line, if any
13:     end if
14:   else
15:     if shape_to IsOfType("Action") then
16:       GenerateCodeAction( shape_to, indentation )
17:       Parse( shape_to, end_condition, indentation )
18:     else if shape_name = "Wait For" then
19:       GenerateStartCodeWaitFor( shape_to, indentation )
20:       Parse( shape_to, end_condition, indentation + 6 )
21:     else
22:       if shape_name = "Everytime" then
23:         endConditions.Add("Everytime_end")
24:         GenerateStartCodeEverytime( shape_to, indentation )
25:         Parse( shape_to, "Everytime_end", indentation + 6 )
26:         GenerateEndCodeEverytime( shape_to, indentation )
27:       else if shape_name = "Switch" then
28:         endConditions.Add("Switch_end")
29:         GenerateStartCodeSwitch( shape_to, indentation )
30:         Parse( shape_to, "Switch_end", indentation + 2 )
31:       end if
32:       if endConditions is not empty then
33:         Parse( shape_endflow, endC, indentation )
34:         where endC = "Stop" if shape_endflow ≠ endConditions.Last() or
            = endConditions.Last() but after removal endConditions is empty
35:         otherwise endC = endConditions.Last()
36:       end if
37:     end if
38:   end if
39:   line ← GetNextLine()
40: end while
```

5.6 Examples

As explained in the previous sections, prototyping an interaction is achieved by connecting together the suitable blocks, setting the appropriate parameters and executing the diagram. Three examples are shown in Figures 48, 50 and 51, which illustrate the creation of an interaction scenario.



Control
Magnetic Line(s)
Name: ML
Add Modify
[✓] Add immediately

Magnetic Line(s)
Points coordinates: x1, y1, z1; x2, y2, z2; etc.
100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0
Pair of points for the line (position in the above list starting from 0 should be used, i.e. 0 1;1 2;2 0:
0 1; 1 2; 2 3

Figure 48: The interaction diagram that prototypes the scenario for creating three magnetic lines, which are removed if the first keyboard key pressed is 'a'.

The first scenario (Figures 48 and 49) was created as an extension of a demonstration from the H3DAPI which simply exhibits three magnetic lines. The lines are first created and when the user subsequently presses the 'a' key, these lines are removed. The interaction scenario (Figure 48) can be read as "create and add three magnetic lines with the given coordinates and order; wait for a keyboard key press, if the key pressed is equal to the value 'a', then remove the magnetic lines, otherwise do nothing". The blocks that have been used, other than the *Start* and *Stop* shapes (the green and red traffic lights) are: the *HapticEffect* block that allows the creation of haptic effects including magnetic lines, the *Wait For* block to listen to one event occurrence, in this case the keyboard being pressed, the *Switch* block to test which key has been pressed, and finally the *Trash* block to

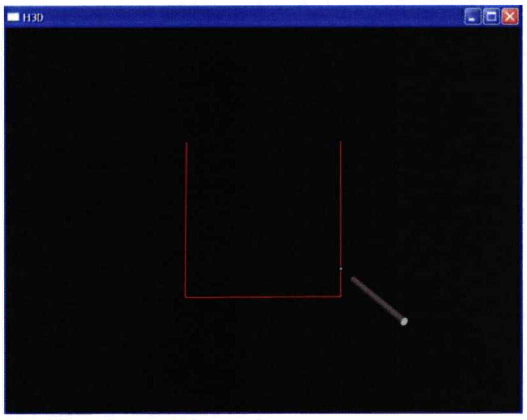
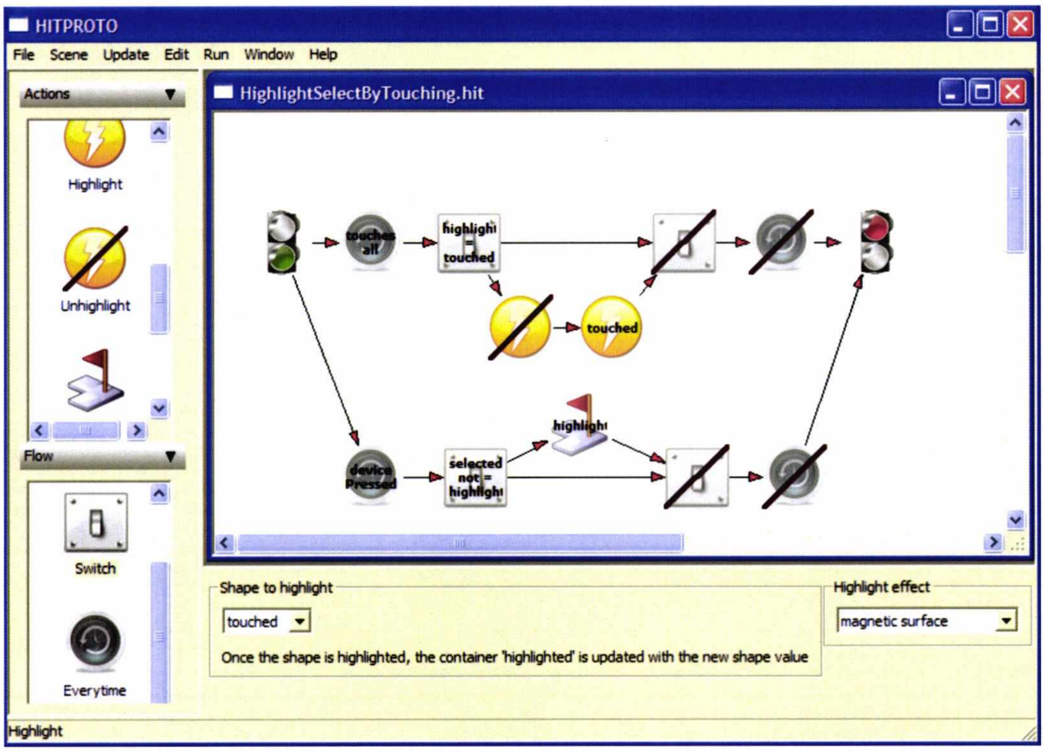


Figure 49: Screenshot of the scene displayed with H3DAPI for the magnetic lines scenario from Figure 48.

remove the magnetic lines from the scene.

The second scenario (see Figure 50) was created in order to reproduce an interaction that has been presented using the language NiMMiT [DBVRC07]. In this interaction scenario, collisions with the objects are monitored and every time the device touches a scene object, if the touched object is not highlighted, then the previously highlighted object is unhighlighted and the touched one is successively highlighted. Haptic highlighting can include creating a magnetic bounding box, adding a spring to the centre of the object or making the object's surface magnetic. In addition, every time the button on the device is pressed, the current highlighted object is selected if it is not already selected.

In the context of haptic data visualization, scenarios including guidance metaphors such as the 'Museum Tour', which conveys an overview of a line chart to visually impaired people; and the use of a repulsive force based on point concentration to convey a 3D overview of scatter plots have been developed using the tool [PRR09]. These are described in more details in the following chapter, in Section 6.1. A simpler version of the 'museum tour' [PR07], which deals with the movement interaction only, is illustrated in Figure 51. This guidance tour takes a user along a predefined path, stops them at predetermined points of interest, letting them freely roam around to get a feeling for the surroundings, and then, after a period of time, the tour continues. This example involves the guidance blocks (*Guidance_Add* to define the guidance settings and *Guidance_Control* to control the movement, either starting, pausing or resuming) and all of the flow blocks (*Wait For*, *Everytime* and *Switch*). The sequence can be read as: create



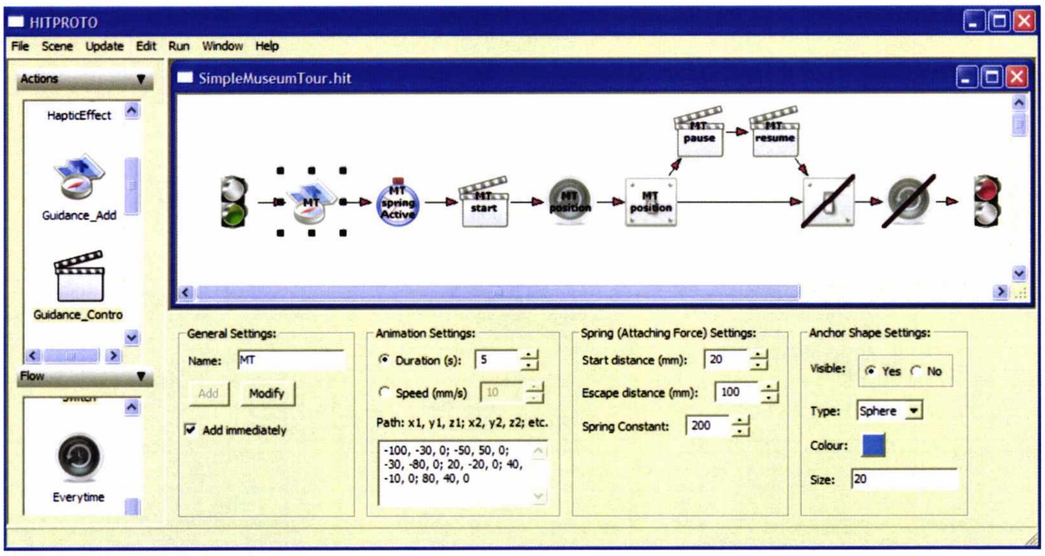
Shape to highlight: touched

Once the shape is highlighted, the container 'highlighted' is updated with the new shape value

Highlight effect: magnetic surface

Highlight

Figure 50: A highlight by touching and select after the device button is pressed scenario, similar to DeBoeck et al. [DBVRC07].



General Settings:
Name: MT
Add Modify
☑ Add immediately

Animation Settings:
● Duration (s): 5
○ Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 200

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 20

Figure 51: The museum tour interaction diagram: the user is led along a predefined path and at chosen points of interest, the tour is paused, to allow the user to explore the surroundings for a period of time before the tour resumes.

and add a guidance object with the specified parameters, wait for its inner spring to be activated when the device is attached and start the guidance; then every time the device is at a point of interest, pause it and allow the user to roam around for a given time before resuming the guidance.

As the examples have shown, HITPROTO currently supports guidance interactions and the addition of haptic effects. HITPROTO is still at the prototyping stage and more interactions will be integrated, starting from interactions for haptic visualization, as described in [DFAD09] to a more complete set of haptic interactions [DBRC05]. As a limited set of interactions is applicable to the haptic modality [DBRC05], the case-by-case approach adopted has been judged appropriate.

5.7 Summary

This chapter described the design and implementation of HITPROTO, a haptic interaction technique prototyping tool, which aims to facilitate the development and testing of haptic interactions, in particular for data visualization. The tool targets non-programmers, such as blind students' teachers and designers, and was therefore developed with the attempt of being simple and intuitive. Consequently, it is based on a graphical dataflow language, which uses configurable blocks to specify an interaction. The blocks include 'action' blocks (e.g. creating, adding, removing guidance or haptic effects) and 'flow' blocks to control the behaviour of the interaction through events or satisfied conditions. The design choices avoided as much as possible abstract and technical concepts, such as input/output linking or variables, in favour of a language close to everyday natural language.

Simple examples introduced the different blocks provided by the tool and illustrated how to use HITPROTO to prototype interactions. More complex examples are presented in the following chapter as well as a qualitative study to evaluate the usability of the tool with non-programmers.

Chapter 6

Case Studies and Evaluation

Evaluation is an important step in the development cycle of any software. Therefore, two methods were used to evaluate HITPROTO. First, to demonstrate the breadth of techniques that could be implemented using the tool, two case studies and novel haptic visualizations were created. Second, to evaluate the usability of the tool, i.e. how easy the tool is to use in practice, an in depth evaluation with nine participants was performed.

Section 6.1 discusses the various case studies, which demonstrate that different and novel haptic visualizations can be created using HITPROTO. Section 6.2 describes the conducted usability testing, and in particular, details the test procedure, the results and the suggested improvements.

6.1 Case Studies

6.1.1 Line Chart Scenario

Line charts are one of the most common representations for statistical data. However, many challenges still remain for their exploration with non-visual techniques. Researchers resort to solving these using the auditory modality (as explained in Chapter 2 Section 2.2.1), but this modality may not be appropriate or available, and so a pure haptic technique would be preferable as an alternative. The challenges consist of getting an overview of the chart and locating and comparing specific features (such as minimum and maximum values, intersections). Getting an overview haptically is difficult, especially when using point-based devices.

Guidance coupled with free exploration can contribute to building a better mental image of the chart (see the discussion in Section 3.1, Chapter 3). To that effect, three different guidance tours have been developed with the line chart application discussed in Chapter 3. These guidance tours have been reproduced using HITPROTO in order to demonstrate that it can help prototype haptic data visualizations. For all of the following guidance tours, the 'VizModule' from HITPROTO was used. As explained in the previous chapter, this module allows the creation of a haptic line chart from a set of points given as input by the user. The generated representation is the same as the representation chosen for the line chart application, and can be a V-shape or a U-shape line with embossed cylinders for axes. The representation is created in an X3D file and the line points, as well as common points of interest, are provided in an output text file.

Museum Tour

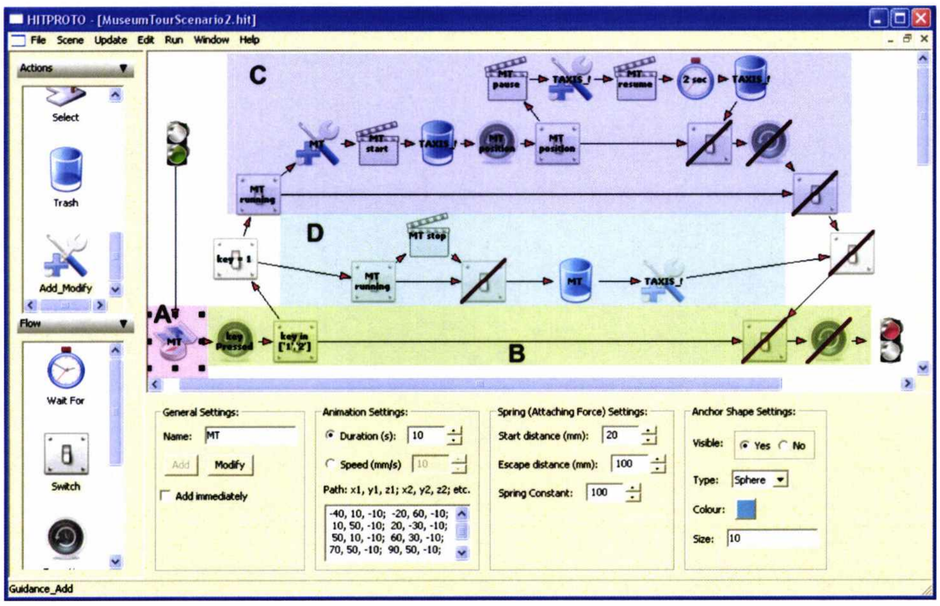
As described in Chapter 3, The museum tour guides a user along a given path and lets them explore the surroundings by pausing at predefined points of interest for a given time before resuming (see Figure 28). This museum tour scenario was reproduced with HITPROTO and Figure 52 shows the corresponding dataflow diagram.

The museum tour interaction is the combination of a standard guidance block and the behaviour at points of interest during the movement. The diagram can be read in two stages: creating the object for the guidance (part A) and specifying the behaviour of the guidance interaction (part B, C and D on Figure 52).

In part A, the guidance object is created using the *Guidance_Add* block. This block enables the user to set general parameters, such as the parameters for the attaching force, the path and speed or duration (see Figure 52, the *Guidance_Add* block is selected and therefore the parameters are displayed).

In parts B, C and D the behaviour of the guidance interaction is specified. The implemented scenario gives the user the choice between two modes of exploration: the museum tour by pressing key '1' (part C) and free exploration by pressing key '2' (part D). These modes thus depends on the keyboard key pressed, which is monitored using the *Everytime* block set to listen to keyboard events and the *Switch* block checking whether the key pressed belongs to the exploration keys (see part B).

In part C (see Figure 52), which corresponds to the key being '1' pressed, the museum tour is started, using the *Guidance_Control* block set to 'Start', if it is not already running, which is checked with the *Switch* block set to the condition "if guidance MT is running is False". If the guidance is already running and the



General Settings:
Name: MT
Add | Modify
☐ Add immediately

Animation Settings:
(●) Duration (s): 10
( ) Speed (mm/s): 10
Path: x1, y1, z1; x2, y2, z2; etc.

```
-40, 10, -10;  -20, 60, -10;
 10, 50, -10;  20, -30, -10;
 50, 10, -10;  60, 30, -10;
70, 50, -10;  90, 50, -10;
```

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 100

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 10

Figure 52: Diagram showing the dataflow of the museum tour scenario in the case of a haptic line chart visualization. The tour stops at all the points of interest: maximum, minimum and intersection with the axes.

user presses key '1' again, then nothing happens. After the guidance is started, the behaviour at the points of interest is specified with the block *Everytime*, *Switch* and *Guidance_Control*. The operation of the *Everytime* block is to monitor the position of the guidance object, and combined with the block *Switch*, which tests whether the movement position is equal to a point of interest, enables the program to decide whether the user is passing through a point of interest and thus some actions should be performed. In this scenario, the desired behaviour is a 'pause-resume' behaviour, i.e. to pause at these points and allow the user to explore the surrounding area before resuming on the original path after a given time. The method to 'pause' and 'resume' the guidance is achieved with two *Guidance_Control* blocks; the first to specify the action of pausing, is set to the 'roaming around' option to allow for a wider range of movement and the second allows the guidance to be resumed after a chosen time. Additionally, the axes are removed during the guidance and added back whenever the guidance is paused.

In part D (see Figure 52), which corresponds to the key being '2' pressed, the free exploration is triggered. If the guidance is running, then the guidance is stopped with the *Guidance_Control* set to 'Stop'; it is removed with the *Trash* block. Finally, the axes are added again with the *Add_Modify* block. The guidance object is removed in the exploration mode, without checking whether a guidance was added. This is because the sensor monitoring when a guidance has become inactive (meaning the guidance has ended), has not been implemented yet. Therefore, the guidance cannot be removed when it has finished unless it is removed for every case (at the programming level, the removal is done only when a guidance instance has been added to the scene graph, but at the high level this additional test is not necessary). When the sensor to monitor the end of the guidance is implemented, then the scenario can be modified to wait for the end of the guidance to remove it (after its addition and start) and therefore stop and remove the guidance object only in the case it is running.

In 'natural' language, the diagram scenario on Figure 52 reads as:

- first create the guidance object with the given parameters;
- then, every time the keyboard is pressed, check if the key is equal to either '1' or '2';
- if the key is equal to '1', check whether the museum tour is running, if not add the guidance object and start it at the device's position, remove the axes and perform the pause-resume behaviour, as described above, every time the guidance movement passes through a point of interest. The axes are added when the guidance is paused and removed again when the movement resumes.
- if the key is equal to '2', then free exploration has been chosen. Therefore, the guidance is first stopped, then removed and the axes are added back. With free exploration, nothing happens, letting the user fully explore the chart representation.

Bus Tour

The bus tour guides a user along a predefined route and constrains the user to that path during the movement (see Figure 27(a) in Chapter 3). This bus tour was reproduced with HITPROTO and Figure 53 shows the corresponding flow diagram.

The bus tour consists of a standard guidance, which is a moving spring to which the user is attached to, and a magnetic line that constrains the user to the chart line. The diagram can be read in two stages: creating the objects for the guidance and the magnetic line (part A) and specifying the behaviour of the guidance interaction (part B and C on Figure 53).

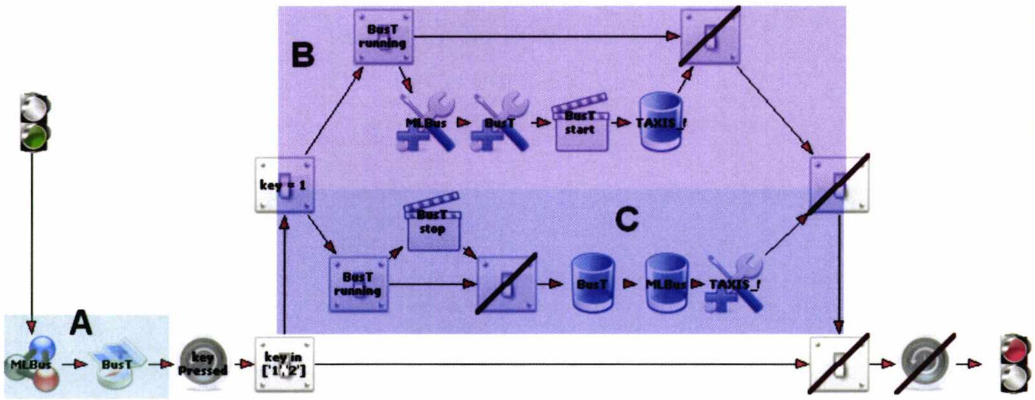


Figure 53: Diagram showing the dataflow of the bus tour scenario in the case of a haptic line chart visualization. Part A setups the guidance object and magnetic lines, part B starts the guidance when appropriate and part C sets the free exploration mode.

In part A, the magnetic lines are created with the first block *Haptic Effect* while the second block, *Guidance_Add* enables the specification of the guidance parameters (such as trajectory, speed, and the pulling force).

In parts B and C, the behaviour of the interaction is defined: if the user presses key '1', then if the bus tour is not already running, it is added and started (part B); if key '2' is pressed instead, then the free exploration is activated (part C). In part B, adding the bus tour is done using two *Add_Modify* blocks set to add the magnetic lines and the guidance object. The bus tour is then started using the *Guidance_Control* block set to 'Start', only if it is not already running. As during exploration, the presence of the axes can impede the guidance movement, the *Trash* block is added to remove them while the guidance is running. If the key '1' is pressed again during guidance (checked with the *Switch* block set to guidance running), then nothing happens (see part B on Figure 53). If key '2' is pressed, then the user changes to free exploration mode (part C). The guidance is stopped with the *Guidance_Control* set to 'Stop' if the guidance is running, the guidance object and the magnetic lines are removed with two *Trash* blocks and finally the axes are added again with the *Add_Modify* block (see part C on Figure 53). The guidance object and the magnetic lines are removed in all cases for the same reasons explained for the museum tour.

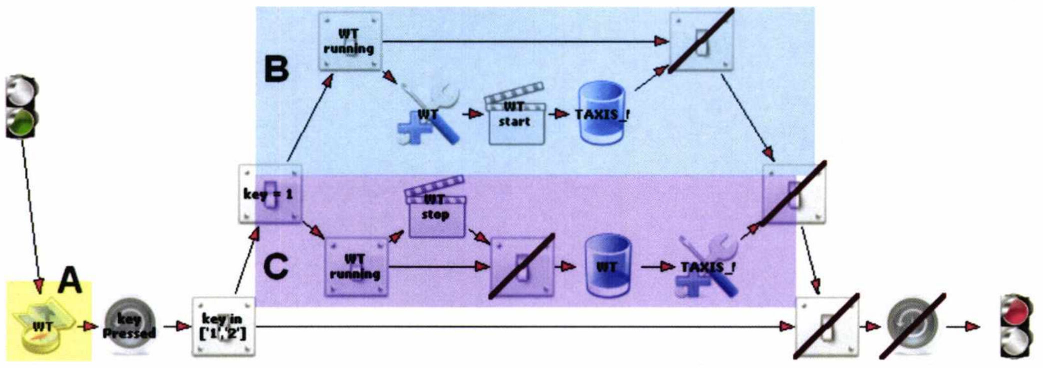


Figure 54: Diagram showing the dataflow of the water skier tour scenario in the case of a haptic line chart visualization. Part A setups the guidance object, part B starts the guidance when appropriate and part C sets the free exploration mode.

Water Skier Tour

The water skier tour is less constrained than the bus tour and simply takes a user along a given route and lets them move from side to side (see Figure 27(b) in Chapter 3). This is achieved with a simple guidance interaction and no further constraints. Figure 54 shows the corresponding flow diagram. The diagram can be read in two stages: creating the object for the guidance (part A) and specifying the behaviour of the guidance interaction (part B and C). In part A, the *Guidance_Add* enables the specification of the guidance parameters (such as trajectory, speed, and the pulling force). In parts B and C, the behaviour of the interaction is defined: if the user presses key '1', then if the water skier tour is not already running, it is added and started (part B); if key '2' is pressed instead, then the free exploration is activated (part C). In part B, adding the water skier tour is done using an *Add_Modify* block and starting it using the *Guidance_Control* block set to 'Start', only if it is not already running. The *Trash* block is used to remove the axes while the guidance is running. If key '2' is pressed, then the user changes to free exploration mode (part C) and the guidance is stopped with the *Guidance_Control* block if it is still running and removed with the *Trash* block, and the axes are added again with the *Add_Modify* block (see Figure 54).

One additional idea to convey information about the points of interest, and implemented for both the water skier and bus tour in the line chart application, is to change the speed of the tour at these points to indicate their presence, by slowing down for example. However, this idea is left for future work and is not integrated in the current implementation of HITPROTO.

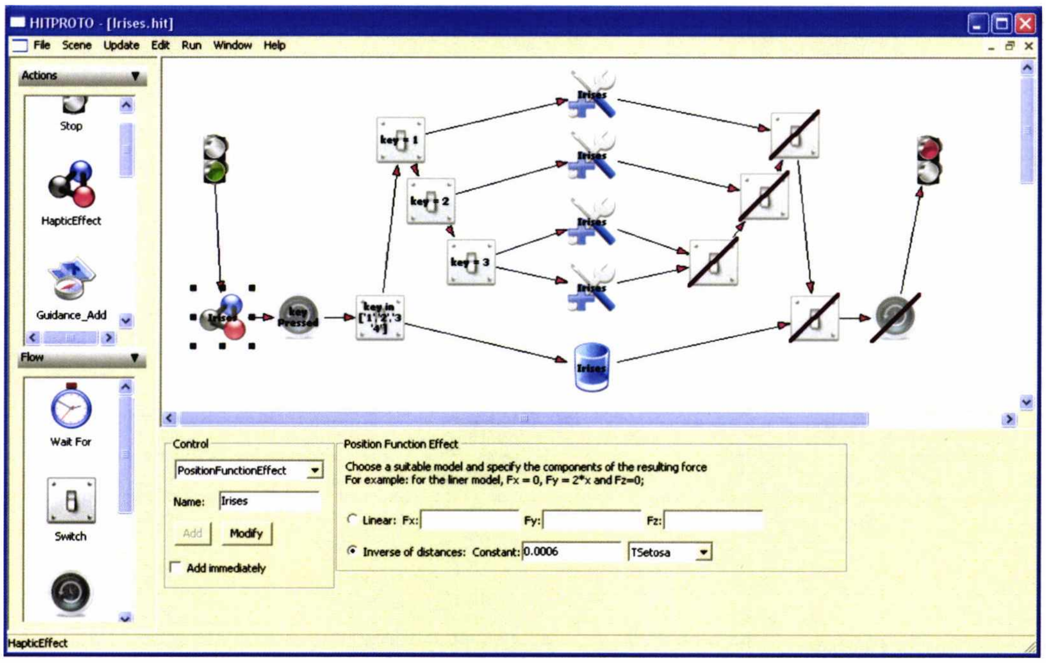
6.1.2 Scatter Plot Scenario

A scatter plot is a visualization technique used to reveal correlation between variables. By displaying data as a collection of points, the scatter plot shows the relationship between variables through the size and location of the point cloud, the direction of the relationship, and whether outliers exist. Analyzing a scatter plot typically consists of two tasks: identifying the general trend (direction, size and position) and spotting interesting features such as outliers. This process corresponds to the visualization mantra of Ben Shneiderman namely "Overview first, zoom and filter, then details on demand" (Chapter 14, page 539 [SP09]). Getting an overview is therefore the first step to understanding the data. Hence a novel overview haptic interaction technique for scatter plots was designed that conveys the general trend of the plot.

The interaction technique was developed using HITPROTO. Figure 55 shows the dataflow diagram of the scenario for the haptic visualization of a scatter plot. The data used was the Iris dataset by R.A. Fisher [Dat10]. Three species of Iris flower (Setosa, Versicolor and Virginica) have been studied, with regard to the following parameters: sepal width, sepal length, petal width and petal length.[1] Both 2D and 3D charts were generated to highlight the correlation of the flowers sepal length and petal length/width (see Figure 56). The concept of this scenario is simple, each dataset is associated with a key on the keyboard ('1','2','3') and pressing that key 'adds' the haptic effect to the corresponding dataset (see Figure 55). The user can then feel the datasets successively as well as the whole dataset. In doing so, the user should get a general idea about the location of the different datasets relative to each other as well as their respective size as it provides simplified and different views of the data [RFC02].

The force model used is part of the PositionFunctionEffect, in the *Haptic Effect* block in HITPROTO, where the user can either specify the 3D components of the force or use a predefined force model, which can be applied to grouping nodes in the scene graph, according to the device's position. For the scatter plot scenario, the predefined model was used (see Figure 57 and 58), which computes the resultant repulsive force to be applied as the sum of the inverse of the distances from the haptic device to each point from the point cloud in the chosen grouping node along with the sum of the unit vectors between the device and these points (see Equation 1 and Figure 57 for an example).

[1]The Iris dataset was obtained from the XmdvTool website, in the data sets category[Xmd10].



Control

PositionFunctionEffect

Name: Irises

Add Modify

Add immediately

Position Function Effect

Choose a suitable model and specify the components of the resulting force
For example: for the liner model, Fx = 0, Fy = 2*x and Fz=0;

Linear: Fx: Fy: Fz:

Inverse of distances: Constant: 0.0006 TSetosa

Figure 55: Diagram showing the dataflow of the haptic scatter plot visualization of Irises. The haptic effect is first created, then when a key is pressed, the haptic effect is either set for a particular grouping node (for all of the Irises or each of them) or removed depending which key was pressed.

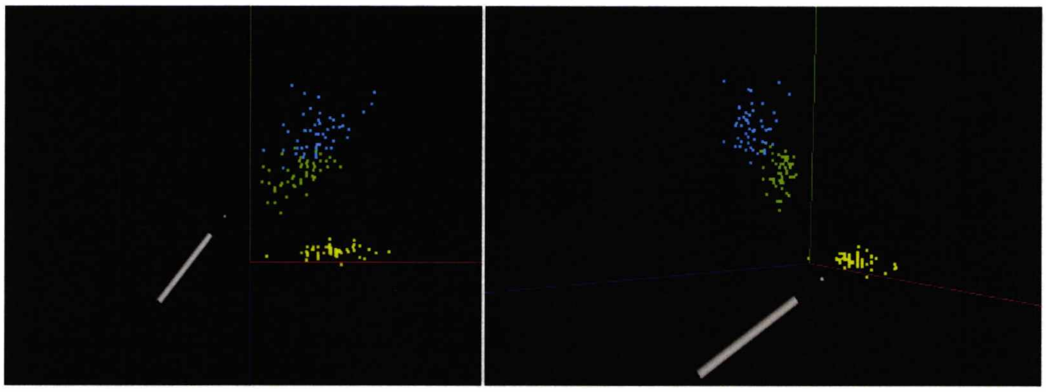


Figure 56: Screenshot showing the 3D visual display of the enhanced Iris datasets used for the haptic visualization as well as a rotated view.

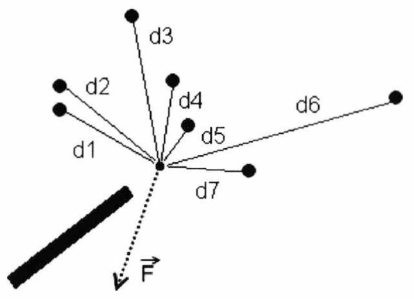


Figure 57: Diagram describing the force model used in the haptic scatter plot visualization scenario. The resultant force is given in Equation 1 with d_i and $\vec{u_i}$ computed for the seven points.

$$\vec{F} = -k \times \sum_{i=1}^{n} \frac{1}{d_i} \times \vec{u_i} \ . \tag{1}$$

with d_i, the distance from point $_i$ to the device and $\vec{u_i}$, the unit vector of the vector from the device to point $_i$.

The initial idea to convey the scatter plot's general trend was a function that depends on the number of points within a given radius. However, simply using the number of points as a factor led to jerky forces thus highlighting the need for a continuous force that would give a smoother feeling. After experimenting and discarding different models (including one trying to use the device's velocity vector), all the points of the dataset were used, and not only the ones present in the device's tip vicinity, along with the continuous model based on the inverse of the distances from each point to the device. The idea was that the closer the device would get to a highly point-concentrated area, the greater the force factor would be, while being further away from the points reduces the force (see Figure 58). This model can convey relatively well the direction of the point cloud and relative position to the other datasets (when felt successively as in the scenario presented above and in Figure 55).

However, it is difficult to get an accurate picture of the point cloud density. Moreover, even though many points were used with these datasets, this technique could become computationally expensive in the case of thousands of points. Lastly, this technique does not represent the axes haptically, and therefore it is not possible to get a general idea of the absolute position of the datasets. Nonetheless,

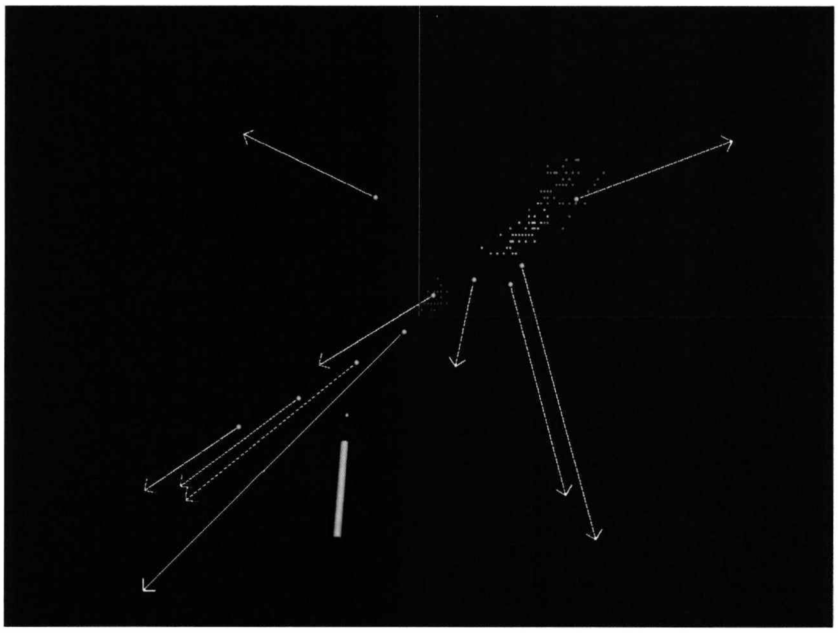


Figure 58: To illustrate the forces on the device, this figure gives a 2D representation of the force model using a 2D Iris dataset. The larger filled circles show the device positions and the vectors show the force applied on the device. The magnitude of the force was scaled down for the figure.

the axes could be included using embossed cylinders or using a guidance interaction technique, as detailed in Chapter 3 and the previous section, by taking the user to the centre of the axes and then to the first point(s) of the point cloud to help the user locate the point cloud within the world coordinate system.

6.1.3 Summary

This section has presented several interaction techniques that aim to help the user get an overview of chart data. Each of the implemented techniques demonstrate different functionality of HITPROTO and confirm that a diverse range of solutions can be implemented using HITPROTO.

The first techniques describe guidance and tour interactions, in particular a museum tour metaphor, in the context of line chart data. The user is taken along the V-shaped line and can explore the surroundings at points of interest, such as minima/maxima and intersections with the axes, where the tour pauses for a given time. Different variations of the guidance (water skier tour, bus tour, museum tour) can be used to provide different types of information (the whole line overview, the points of interest or the surroundings).

The last interaction technique displays scatter plot data and uses a force model based on the sum of the inverse of the distances to each point from the current user location to feel the data. This interaction scenario allows the user to feel the different datasets successively, thus providing the user with the relative positions of the datasets and their general direction.

6.2 Evaluation of HITPROTO

This section presents the different stages and results for the usability evaluation of HITPROTO. First, the choices for the type of testing method and the number of participants are explained in Section 6.2.1, based on a survey of usability literature. Then, the selected testing method and the general procedure are detailed in Section 6.2.2. The pilot study conducted with two participants and its results are subsequently described in Section 6.2.3. The pilot study revealed some issues with the current design of the materials, such as the duration of the study and the length and difficulty of materials. Therefore, the required changes to the testing method and materials are listed in Section 6.2.4 and the new conditions for the full experiment are summarized in Section 6.2.5. The results of the conducted experiment are analyzed in Section 6.2.6 and the recommendations that emerged from the study in order to improve the usability of the tool are listed in Section 6.2.7. Issues in the system highlighted by the study required changes to the implementation. These are described in Section 6.2.8.

6.2.1 Usability Evaluation

Given the stage of development and the importance of getting some user feedback, usability testing has been chosen to evaluate the HITPROTO prototype. Usability refers to how well and easily users can learn and use a product to achieve their goals and how satisfied they are with that process [Usa10]. Usability is a combination of attributes, also referred to as usability metrics, and these are not exactly agreed on as several definitions are available ([Rub94], chapter 1, p18-19, [Usa10], [Nie03], [IUS01]). However, all these definitions encompass similar concepts under different terms, including: "The accuracy and completeness with which users achieve specified goals", often referred to as effectiveness; "The resources expended in relation to the accuracy and completeness with which users achieve goals", often referred to as efficiency; and "Freedom from discomfort, and positive attitudes towards the use of the product", often referred to as satisfaction [IUS01]. Moreover, even though the definitions of the metrics differ, they point towards the same measurements such as the error rate, the task success rate and the completion times.

It seems commonly agreed in the literature [Usa10, Rub94, IUS01] that usability testing can be divided into two main categories or approaches. These are referred to as formative and summative tests [IUS01, SP09] or as diagnostic and formal tests [Usa10, Rub94] or simplified to qualitative and quantitative tests. Formative or diagnostic testing focuses on gaining insights into the problems that arise when using the interface in order to improve it, while summative or formal testing aims to measure statistically the interface against usability metrics, to validate or refute some hypotheses.

For a first evaluation, with a focus on evaluating whether the participants can use the tool to perform the tasks and on finding the improvements to be made to ensure successful use, formative testing was selected as the most appropriate method of assessment. More precisely, the 'assessment test' as defined by Rubin [Rub94] was chosen. An assessment test "seeks to examine and evaluate how effectively the concept has been implemented. Rather than just exploring the intuitiveness of a product, [one is] interested in seeing how well a user can actually perform full-blown realistic tasks and in identifying specific usability deficiencies that are present" [Rub94] (Chapter 2, p38). In this evaluation both qualitative and quantitative measures (task completion rates) will be collected; as suggested by Rubin [Rub94], "Assessment tests usually gather both types of measures" (Chapter 7, p156).

After finding the suitable type of usability testing, choosing a suitable number of participants becomes an important issue. Therefore, for a long time, usability research has debated the minimal sample size for optimizing the cost of usability studies. A controversy started when a statistical formula, that exhibited that five users were enough to exhibit 85% of all usability problems in a product, was introduced by Nielsen [NL93, Nie00] and others. Nielsen states that "The best results come from testing no more than five users and running as many small tests as you can afford". This claim relies on the assumption that finding the problems is independent of whether they were found before and that the problems are independent of each other. Many studies have since been conducted to assess the validity of this claim. The question is yet not agreed upon, dividing the usability community into a group in favour of Nielsen's claim [Lew06, MWT$^+$02] and those against [SS01, WC01, Fau03]. Those in favour advocate that the most serious problems are found with the first few participants [Nie00] thus allowing to test and applying fixes rapidly and iteratively. On the other hand, critics have shown that the formula only applies for specific and often simple cases and that the most severe problems are not always found by the first testers.

However, this debate fails to specify to which category of user testing it applies to. Katz and Rohrer [KR04] underlines that "Determining how many participants in a study is 'enough' is a question that is appropriate when the goal of the usability study is one of assessment". Indeed, in terms of statistical analysis, summative tests, which focus on achieving significant statistical results, need many participants and the minimum number needed for significance is important. In this case, Nielsen's claim concerning five users is obviously not applicable and this is why he did not make it for this type of testing [Nie06]. The controversy thus stems from a claim concerning formative or qualitative testing. Nonetheless, as the formative testing emphasizes the improvement of the interface, this debate seems irrelevant. Several iterations with a small sample size and refinement in between the testing iterations is more beneficial than one iteration with a large number of users when the goal is to improve the interface and fix as many problems as possible [KR04, SP09].

Recently, Lindgaard and Chattratichart [LC07] demonstrated that the number of users does not significantly correlate with the number of problems discovered but that the task coverage does. Therefore they concluded that the endless debate on the sample size was useless and instead task coverage and its impact on problem discovery should be more thoroughly investigated.

Even though HITPROTO could be tested with as few as five participants with several iterations, due to timing constraints that prevented conducting several test iterations, Rubin [Rub94]'s recommendation of using eight participants was followed and in the end, nine participants were used for the evaluation. Also, the evaluation tasks were chosen to cover most of the functionalities in the tool (as suggested by Lindgaard and Chattratichart [LC07]).

6.2.2 Test Plan

The main usability test procedure is detailed in the following paragraphs; followed by the details and the results of the pilot study, and then the full study.

Purpose

The main purpose of the test is to assess whether a user can use the tool to prototype haptic interactions. Additionally, it will be observed whether the tutorial helped learning sufficiently so that users could operate the tool.

Problem Statements - Test Objectives

Documentation/Tutorial:

1. Is a more detailed manual needed other than the tutorial?
2. Is online-help also required?

Tool:

1. Are users able to interact with the blocks easily (drag'n'drop, moving blocks on the canvas, linking them, accessing the bottom information of a block)?
2. Can users perform common tasks easily and accurately?
3. Are there some major usability flaws or obstacles that prevent completion of the most common tasks?
4. Could users learn enough to use the tool in the time allocated during the testing?
5. Could users control the bottom panel easily to set the parameters of the blocks?
6. Did the blocks correspond to the functionality the user expected?
7. Once a diagram was completed, could the user generate and execute it with the API easily?

User Profile

Due to the difficulty of recruiting designers or teachers for visually impaired students, postgraduates from the University of Kent will be recruited instead, on the assumption that the tool should be accessible to any person with no programming knowledge, no matter their background, and that any of these postgraduates could be potential users in the future. The profile of the participants for the experiment has been summarized in Table 6.

Characteristic	Range
Gender	Female/Male
Age	18-65
Education Level	Postgraduate
Subject of study	Anything but Computer Science
General computer experience	Several years (can use a computer and GUI-based interfaces)
Programming experience	None to beginner
Haptic experience	None to interaction with market products with limited integrated haptics (such as in gaming, tactile phones)
Visual programming tool experience	None to limited

Table 6: Participants profile

Methodology

The usability test will consist of a single performance test designed to gather usability data via direct observation. The test is composed of the following four sections:

1. *Participant greeting and background questionnaire:* Each participant will be personally greeted by the test monitor and made to feel comfortable. The participants will be asked to fill out a short questionnaire that gathers basic background information. This questionnaire is available in Appendix B.2.

2. *Orientation:* The participants will receive a verbal scripted introduction and orientation to the test, explaining the purpose and objective of the test and additional information about what is expected of them. They will be reminded that the tool is the focus of the evaluation and not themselves, and that they should perform in the way that is typical and comfortable to them. The participants will be informed that they are being observed. This experiment script is included in Appendix B.1.

3. *Prerequisite Training:* After the explanation of the test procedure, and before starting the actual performance test, the participant will be introduced to the haptics technology through a few demonstrations and then will undergo a prerequisite training to acquire basic skills to use the tool and ensure the performance test assesses the ease of use rather than the ease of learning. The training session will also be monitored to get insight into the ease of learning. The learning objectives include the participant being able to create

a new interaction diagram, manipulate the blocks and tune their parameters (drag'n'drop and selection of values from the bottom panel), connect these to create the interaction scenario, compile and execute the interaction diagram and test whether it achieves the given interaction scenario goals. All of these tasks should be achieved for simple task scenarios that will also introduce the different blocks used in the performance test. The training will be provided in the form of a 'getting started' tutorial which includes various interaction scenarios to walk the participants step-by-step through the prototyping examples, and in the mean time introduces the different components of the tool that would be used in the performance test. The tutorial ends with the criterion test, in the form of a 'check yourself' task scenario example, the participant should be able to achieve on their own before starting the performance test. This criterion test aims at evaluating whether the participant has understood how the tool works and is able to proceed with the performance test. The solution was provided at the end as well as help on demand.

4. *Performance test:* The performance test consists of a series of tasks that the participant will be asked to carry out while being observed. The scenario is as follows: After the training is complete, the participant will be asked to complete a set of four common prototyping tasks. The participants will be encouraged to work without guidance. The test monitor may ask a participant to verbalize his or her thoughts if the participant becomes stuck or hopelessly confused. This will help to pinpoint the reason for the problem and will be noted by the test monitor. During the main performance test, elapsed time (and errors) will be noted for each task on the task list. The test monitor will also make notes about relevant participant behaviour, comments and any unusual circumstances that might affect the result (e.g., the tool malfunctioned or crashed).

5. *Participant debriefing:* After all tasks are complete or the time expires, each participant will be debriefed by the test monitor and the debriefing session recorded. The debriefing will include the following:

   - Filling out a preference questionnaire pertaining to subjective perceptions of usability and aesthetics of the tool
   - Participant's overall comments about his or her performance

- Participant's responses to probes from the test monitor about specific errors or problems during the test, if there were any

The questionnaire and the debriefing form are detailed respectively in Appendix B.5 and B.6. After the debriefing session, the participants will be thanked for their effort and then released. They will be paid five pounds an hour for their contribution.

Task List

The four evaluation tasks will consist of prototyping scenarios and will involve subtasks such as choosing the right blocks for the interaction, drag'n'dropping them onto the diagram, connecting them together in a logical way and tuning their parameters to obtain the required behaviour and executing the produced scenario to test whether the task is completed or not. For all the tasks, the successful completion criteria will be whether the behaviour described in the task scenario has been achieved by the interaction diagram 'programmed' by the participant. For each task, the participant will be provided with the software running and the haptic setup to test the interactions. The scenarios will span the range of easy to more difficult scenarios and involve all the functional blocks.

Test Environment and Equipment

The environment will be an office, including a desk, computer, chair, typical supplies such as pencils, pens and so on. On the desk there will be a PC attached to the haptic setup which includes a monitor, a glass and a haptic device, the PHANTOM force-feedback device. The prototype must be in working order and be able to accomplish all functionality required by the tasks on the task list.

Test Monitor Role

The test monitor will sit in the room with each participant while conducting the test. The test monitor will initiate the training and later the evaluation tasks and record timings, errors and observations. The test monitor will not help any of the participants unless a question about the test procedure arises. Participants will be asked to rely on the tool, its tutorial, and their own abilities to perform the required tasks. However, if the participant gets stuck during a task, after first encouraging the participant to think more, the test monitor will probe for the reasons of the difficulties and give hints if necessary to see whether assistance leads to task completion.

Evaluation Measures

The following evaluation measures will be collected and calculated:

1. The average times to complete each task across all participants.
2. The percentage of participants who finished each task successfully, the percentage of participants who finished each task successfully with assistance and the percentage of participants who had errors from which they could not recover.
3. Error classification: to the degree possible, each error will be classified and a source of error indicated. Error classes are as follows:
    - Observations and comments - the test monitor notes when participants have difficulty, when an unusual behaviour occurs, or when a cause of error becomes obvious.
    - Noncritical errors - an individual participant makes a mistake but is able to recover during the task.
    - Critical errors - an individual participant makes a mistake and is unable to recover and complete the task.
4. Participants rankings of usability of the tool

6.2.3 Pilot Study

The pilot study followed the main procedure described above, in Section 6.2.2. The tutorial described three interaction scenarios that would introduce all the blocks in the tool. The first scenario was the magnetic lines example presented in Section 5.6, the second a simple scenario to start a guidance interaction after a button press, while the last one was the 'highlight by touching' metaphor also described in Section 5.6. The 'Check Yourself' example, which was a variation of a guidance interaction, was used as a criterion task, which would evaluate whether the participant should continue with the tasks or needed more explanation. The tutorial used in the pilot is presented in Appendix C.

The participants were given four interaction scenarios (or tasks) to perform and were told that if helpful, and especially when the scenarios were similar, they could modify the current diagram rather creating a new one. The interaction scenarios are listed below (as presented during the pilot study):

Scenario 1: Create an interaction where the device will be led along a given path by an anchor object. This interaction will only start both after the keyboard key "s" is pressed and the device is attached to the anchor object. If the device is detached from the anchor or a key apart from "s" is pressed, the scenario ends.
The data points for the path are as follows: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).
The rest of the parameters are left to your choice.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario1.hit

Scenario 2: Create an interaction where the device will be led along a given path by an anchor object. This interaction will start after the device has been attached to the anchor object. The interaction should be paused (if it is moving) or resumed (if it is already paused) each time the device's button is pressed.
The data points for the path are as follows: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).
The rest of the parameters are left to your choice.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario2.hit

Scenario 3: This interaction can be seen as a "Museum Tour" metaphor where a visitor is led along a path and stops at predefined points of interest, such as paintings or sculptures, for a given time, before moving on to the next item.
Create an interaction where the device will be led along a given path by an anchor object. The interaction starts as soon as the device becomes attached to this anchor object. Once the interaction is running, each time the movement passes by one of the given points of interest, the interaction is paused and then the user/device is given a wider range of movement for 3 seconds before the interaction is resumed.

The data points for the path are: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).
The points of interest are as follows: (-50, 50, 0; 40, -10, 0).
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario3.hit

(if time)
Scenario 4: Create and add a force that depends on the device's position. The force created will be proportional to the inverse of distances from the device to each point in a given scatter plot and will use the direction vector. This force should be added after each time the key "s" is pressed and removed after each time the key "k" is pressed. Use 0.0006 for the constant value. Details about how to create such a force model are described in the User Guide.
The scatter plot to be loaded is called "SPIrises3D" and is located in the shortcut folder ExperimentData on the desktop and within that folder in the subfolder ExperimentScenes. The scatter plot describes data points for three Iris flower species: Setosa, Virginica and Versicolor, which data is respectively contained in the TSetosa, TVirginica and TVersicolor nodes. TSPGlobal is the global node that contains all the previous nodes.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario4.hit

The difficulty of **Scenario 1** lay in understanding that it is possible to add two *Wait For* blocks after one another, while for **Scenario 2** in finding the right test with the *Switch* block, which had not been introduced in the tutorial. **Scenario 3** is a simplified version of the 'Museum Tour', as explained in Section 5.6, and its difficulty involves getting the right combination of the *Everytime* and *Switch* blocks with the appropriate option, also not presented during the tutorial. **Scenario 4** is a simplified version of the scatter plot overview interaction [PRR09] (described above in Section 6.1.2). The first three task scenarios have increasing difficulty and the last one was performed if time allowed. Two hours were allowed for the test, with one hour for training and one hour for completing the tasks.

Two participants, one male and one female, took part in the two hour pilot study. Both were postgraduate students with no experience with haptics, programming or visual programming tools and were paid 10 pounds for their participation.

Pilot Results

The first participant took most of the two hours for the training phase with the tutorial and consequently did not perform any tasks. When questioned, he could not explain the functionality of the blocks or how to program an interaction. However towards the end, after repeating explanations about the blocks, the participant was able to understand and use them. He commented that "it clicks, after a while", but he needed several repetitions before he understood the concepts and remembered the functionality. He also thought that a longer training period would help. He added that the tool was easy to use and that the difficulty lied in understanding the language, which needed more time and practice.

The second participant took 2h30 to complete the experiment. She finished the tutorial in one hour and exhibited a better understanding of the different concepts. She completed the 'Check Yourself' example with a little bit of help. She then performed the tasks, spending 24 minutes for the first task, 20 for the second and 16 minutes for the last one. She managed to perform these with very little help: she would ask, for instance, if one option (not introduced in the tutorial but available in the user manual) would do what she expected or more explanations about the interaction description. She clearly understood the language behind the tool and had the right logic to program the interactions; however instead of referring to the manual for more details, she would ask for help. During the experiment, she was really excited about the tool and insisted on finishing the three tasks even when we ran out of time.

6.2.4 Changes After Pilot

The pilot study demonstrated two extreme behaviours; one participant not being able to perform the tasks and one able to understand well and complete all the tasks. Despite these differences, both participants highlighted that two hours were not enough to complete the experiment, as even the participant who was able to conduct the tasks took 2.5 hours. Therefore, the experiment was extended to last three hours, with approximately 1h20 for training, up to 1h15 for the test and the rest of the time to fill in the information questionnaire as well as the final questionnaire followed by a short interview.

Also both the tutorial and the tasks were simplified. The tutorial was changed to include two example scenarios instead of three, and a two-page visual summary of the blocks was created, which summarized the functionalities of the blocks. As for the tasks, the first two scenarios were changed to be variations of the tutorial examples, while the last two tasks were the same as the first and third task scenarios of the pilot study. These tasks, as explained in Section 6.2.3, required the user to apply their understanding of the system to the use of new functionalities that were not included in the tutorial (such as being able to use two *Wait For* blocks one after the other, or new block options). The modified tasks for the experiment are listed below:

Scenario 1: Create a square outline which is magnetic. The square should only appear after the device's button is pressed.
The data points for the square are as follows: (100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0).
The rest of the parameters are left to your choice.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario1.hit

Scenario 2: Create an interaction where the device will be led along a given path by an anchor object. This interaction will only start after the keyboard key "s" is pressed and it will start at the device's position.
The data points for the path are as follows: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).
The rest of the parameters are left to your choice.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario2.hit

Scenario 3: Create an interaction where the device will be led along a given path by an anchor object. This interaction will only start both after the device is attached to the anchor object and the keyboard key "s" is pressed. If the device is detached from the anchor or a key apart from "s" is pressed, the scenario ends.
The data points for the path are as follows: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).

The rest of the parameters are left to your choice.
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario3.hit

Scenario 4: This interaction can be seen as a "Museum Tour" metaphor where a visitor is led along a path and stops at predefined points of interest, such as paintings or sculptures, for a given time, before moving on to the next item.
Create an interaction where the device will be led along a given path by an anchor object. The movement starts as soon as the device becomes attached to this anchor object. Once the movement has started, each time the movement passes by one of the given points of interest, the interaction is paused and then the user/device is given a wider range of movement for 3 seconds before the interaction is resumed.
The data points for the path are: (-100, -30, 0; -50, 50, 0; -30, -80, 0; 20, -20, 0; 40, -10, 0; 80, 40, 0).
The points of interest are as follows: (-50, 50, 0; 40, -10, 0).
Once you have finished the task, please save the file in the shortcut folder ExperimentData on the desktop.
The name of the file should be: [yoursurname]_scenario4.hit

6.2.5 Full Experiment Conditions

The experiment was conducted using the general procedure and the modified materials (tutorial, tasks and visual summary) from the pilot study with the time taken for each participant increased as given above. The modified and new materials that were given to the participants are included in Appendix D. Nine participants – three males and six females – all with no or little programming experience and no former experience with haptic devices or visual programming tools, took part in the evaluation. Each were postgraduate students, their age ranging between 22 and 29 years old and with backgrounds including anthropology, archeology, psychology, microbiology and actuarial science.

6.2.6 Experiment Results

Time for Task Completion

The task-completion time was measured for all the tasks. Apart from two participants – one who only completed one task out of four, and another who did not finish the last task – all the participants managed to complete the tasks, with or without help. The resulting completion times averages for each task are summarized in Table 7. The times were averaged over the number of participants who had completed that task (i.e., averaged over nine, eight, eight and seven participants, respectively).

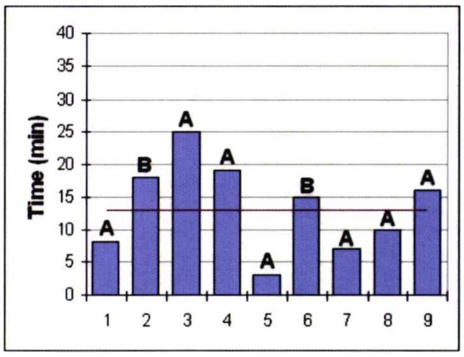
Table 7 shows that most participants completed the tasks within a relatively short period of time. The averaged times show an increasing time for each successive evaluation task. This trend could be explained by the fact that the tasks were gradually getting more difficult. However, at the individual level (see Figure 59 and 60), this trend only appears for three participants with two participants even exhibiting the opposite trend. The opposite trend can be explained by the fact that the more the participants were using the tool, the more they were familiar with it and able to more quickly find a solution. The rest of the participants show no particular trends in their timings. It is also worth noting that the tool crashed for four participants: for participant 1, for the first task (the work was saved and therefore did not affect the performance); for participant 2, during the first task (the work was lost) and the second task; for participant 5, at the end of task 2 (the work was lost) and for participant 9 during the first task (the work was saved and the performance not affected). It is believed that the high values for participant 2 were partly affected by the successive crashes encountered.

	Task 1	Task 2	Task 3	Task 4
Minimum time	3	6	9	14
Maximum time	25	19	23	36
Average time	13	14	18	23

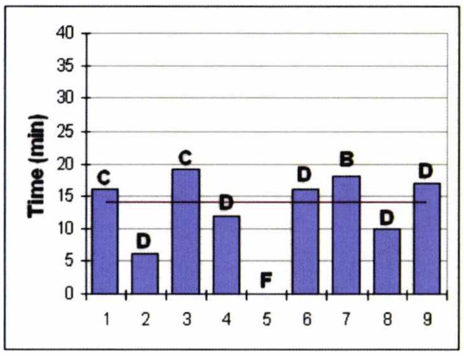
Table 7: Times (in minutes) for participants to complete each task.

Success Rates for Task Completion

The training time was clearly too short to learn the full functionality of the tool; especially as participants needed to learn the visual language, each of the block shapes and their functionalities. Thus, in order to mitigate against this, struggling

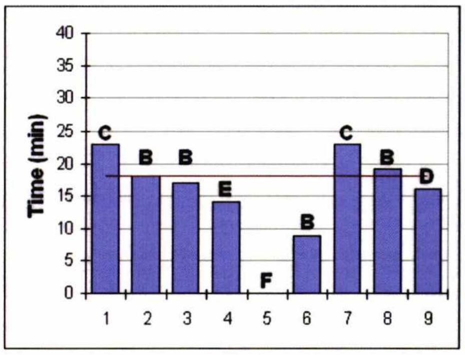


(a) Task 1

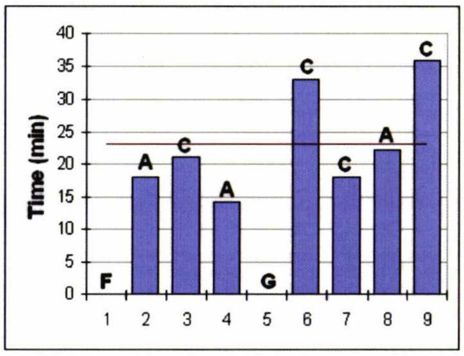


(b) Task 2

Figure 59: Task completion times (in minutes), for task 1 and 2, with the average for each task and the type of success for each participant (see also Table 8 for the meaning of the letter codes).



(a) Task 3



(b) Task 4

Figure 60: Task completion times (in minutes), for task 3 and 4, with the average for each task and the type of success for each participant (see Table 8 for the meaning of the letter codes).

participants were allowed to ask for help. The help given ranged from simple hints, such as *"Refer to page X (or example Y) in the manual."*, to more elaborate hints in the form of questions, such as *"You want to monitor the movement of the guidance? Which blocks allow you to listen and monitor events?"*. The answer was never directly given and after giving some further hints, the participants were left to find the solution on their own.

Table 8 summarizes these results, with the number of participants for each category. The categories include: success without help, success with minor help (i.e. page reference), success with major help (i.e. discussion including questions and explanations), minor errors without help (e.g. starting the guidance at the

	Task 1	Task 2	Task 3	Task 4	Rate(%)
(A) Success no help	7			3	27.8
(B) Success minor help	2	1	4		19.4
(C) Success major help		2	2	4	22.2
(D) Minor errors no help		5	1		16.7
(E) Minor errors minor help			1		2.8
(F) Failure (major errors)		1	1	1	8.3
(G) Not attempted at all				1	2.8

Table 8: Tasks success rates

device's position when it was not asked; but the general behaviour would be correct), minor errors with minor help, failure and task not attempted at all.

The success rates could indicate whether one task was more challenging, unsuitable or impossible. However, Table 8 indicates that most of the tasks were achieved successfully with no or little help, and that more help was required in the latter tasks. This matches well with the design of the evaluation; as the latter tasks were designed to be more challenging than the earlier ones. Also, overall, 88.9% of the attempts at the tasks resulted in a working interaction, with or without help, while only 8.3% of them resulted in failures, despite the help given.

Questionnaire

Table 9 summarizes the main topics mentioned in the questionnaire. Their answers were then discussed during the interview. In general, the participants gave positive feedback. Table 9 shows that most of the participants found the functionalities useful and easy to use, such as displaying the selected parameters on the block, the different interactions within the tool or tuning the parameters with the bottom panel. Eight participants out of nine found the tool easy to use, especially after some practice, responding with comments such as "After some practice and searching quite easy" and "it was easy to use, especially as I have little experience with this type of computer program". Only one participant remarked "I don't know. Similar to other software I guess".

When asked to list aspects of HITPROTO that they liked, the participants answered encouragingly saying "Simple symbols/icons. Easy to understand", "It's easy for a non-programmer to actually program. It's a relatively easy interface. The fact that it is diagram-based", "Intuitive. Could run even when task unfinished, useful to make sure you are on the right lines" and "It made developing/creating things that looked complicated relatively simple. The layout of the

Question	Positive	Negative	Other
Was the tutorial easy to understand?	6	3	0
Did you find using the haptic device difficult?	2	7	0
Did you like the images used for the blocks?	7	2	0
Did the image blocks correspond to the functionalities you expected them to have?	6	3	0
Did you find it useful that the image block displays parameters once they are selected?	8	0	1
Was the bottom panel easy to use to control the parameters?	8	0	1
Were the drag and drop, selection and linking interactions easy to use?	9	0	0
Were there some interactions missing that you would like to be available?	2	7	0
Was the tool easy to use?	8	0	1
Did the tool enable you to prototype and test interactions?	8	0	1
Would you use the tool rather than learning programming?	9	0	0

Table 9: Questionnaire answers. The other column refers to the "I don't know" answer, except for the question concerning the bottom panel, where the answer corresponds to "medium difficulty".

tool made it easier to access".

When asked to list things they did not like about the tool, three participants commented there was nothing they disliked. The others made some suggestions to improve HITPROTO, including "...I was not always sure about the order of connecting icons", "[it] could get messy – a grid to keep the object in place might be useful" or "Cannot zoom out, the start node is always stuck in the beginning, icons". Two participants commented on some of the technical terminology that was used, which they felt could be simplified. One participant simply said that there was "Lots of things to remember".

When specifically asked to suggest improvements, three participants wanted a better help facility, rather than needing to check the tutorial or visual summary. They suggested including a tooltip functionality, where by rolling the mouse over an icon on the diagram a short description would popup. A participant added that an error checking or compilation mechanism would be useful. Although no other participants made that comment, it was observed that many participants committed minor mistakes, such as giving a point index out of range when specifying magnetic lines, or forgetting to choose which instance/object to control or monitor. They were getting confused as their understanding was telling them the diagram was correct, and indeed was, but yet not working, and tried other things before eventually finding out their mistakes. Another participant suggested including "a section where you can see what you have created in 'real time', while creating". One participant repeated that they would like a 'snap-to-grid' functionality. Two other participants were not quite satisfied about the look of the icons and suggested changing the size of the blocks or their name. Three participants had no suggestions to make. Most of these suggestions would be possible to implement; in fact an error checking mechanism could speed up the creation of interaction diagrams.

The System Usability Scale (SUS) [Bro96] was also used in the questionnaire, to evaluate the global usability of the tool. The questions range from the ease of use, to confidence, the tool design, and learning. The average of the SUS scores rates the usability to be about 67%, which is relatively good. All the values are included between 50 and 92.5%, except for one participant who rated the usability to be 17.5%. That participant gave rather low scores overall, and did not seem interested in any computing software (that participant is the one who replied "I don't know" about the ease of use and whether it allowed the prototyping of interactions) and did not want to spend time learning a new tool.

Holistically, including the interview, it can be concluded that the participants enjoyed using the tool and found it relatively easy to use and create dynamic and interactive haptic visualizations. However, four participants complained in the questionnaire that they did not have enough time to familiarize themselves with the tool; they wished to experiment and explore the tool more, such to understand the functionality of each block, before starting the evaluation tasks. They commented: "It was relatively easy to understand, but there was a lot to remember in a short space of time [...], the short space of time meant that I didn't have time to fully understand every element of the tool", "Quite easy but sometimes a bit too fast", "If I had more time to familiarize myself with it, I think it would be very easy to use" and "More time to play around and experiment with the tool and program before starting the task". They all commented, when prompted in the interview, that with a longer training period, they would find the tool very easy to use.

Limitations of the Results

The validity of the results is constrained by two main assumptions. First, the participants are not the typical users. The tool is aimed at visually impaired people teachers, designers and developers new to the H3DAPI, while the participants recruited for the experiment were postgraduate students. Therefore, the first assumption is that although not being the target user group, the postgraduate students can potentially become so in the near future thus being not ideal participants but suitable for a first evaluation.

Second, the participants were provided with some help when they were struggling with the tasks. As the main objective was to find out whether the participants could use the tool and reveal usability issues rather than making quantitative claims, it was assumed that the help given was mostly needed due to the lack of a longer learning and familiarization with the tool period (assumption also believed by most participants after the study). Therefore, as a longer training period could not be afforded, providing help was presumed not to change much the overall usability outcome of the study.

6.2.7 Findings & Recommendations

The usability evaluation showed that, overall, participants could use HITPROTO to prototype haptic interactions. Many of them needed some help; especially as they found that they did not have enough time to learn how to use the tool and familiarize themselves with it. However, these results are promising as people with no or very little programming skills could use the tool and understand the logic behind the dataflow language to program haptic interactions. Furthermore, they managed to do so in a relatively short amount of time with a short training period given the novelty of the tool to them. This reinforces the hypothesis that learning how to operate such a tool would take less time and be more beneficial than learning how to use the haptic API and the corresponding programming languages (such as C++); thus promoting the development of haptic interactions, in particular for haptic data visualization.

The evaluation also highlighted areas for improvement for future work. These include integrating a tooltip functionality for the visual programming blocks of HITPROTO; incorporating an error-checking and compilation mechanism (sometimes some blocks were not well connected or some end blocks missing and it was working, but it was not correct); and changing some terms that may be too technical for a wide audience. Also during the evaluation, a few crashes occurred and a few bugs were revealed. These are now fixed. Also, in future work, the number of interaction techniques available in the HITPROTO prototype will be extended.

6.2.8 Implemented Improvements

The tooltip functionality was suggested by most participants, therefore implying it is an important and almost required functionality. Consequently, following the evaluation, this functionality has been integrated within HITPROTO. When the mouse is hovered over a block icon, a summary of the block functionality is displayed after a few seconds, and disappears again when the mouse is moved. The text used to describe the blocks in the tooltip has been adapted from the blocks summary descriptions, available in Appendix B.4. This functionality enables the user to get a description of the block while constructing the diagram, thus keeping the attention to the screen, instead of checking a printed summary that involves dividing the attention between the screen and the print.

6.2.9 Summary

The primary goal of HITPROTO is to allow rapid prototyping of haptic interactions without requiring prior programming skills. Learning how to use the tool to prototype interactions should take less time than learning how to use the API and the programming languages involved (Python or C++), especially for people with no programming background. This is supported by the evaluation, and although no formal conclusions can be formed, the informal feedback plus the demonstrated ability of many non-computer specialists to create haptic interactions with a relatively short training time, gives us confidence that the tool achieves its goals.

Work on the tool is ongoing. Future work will consist of adding more blocks and investigating more haptic visualization scenarios; in particular for under-researched areas in haptic data visualization such as tables and networks.

Part III

Summary and Conclusions

Chapter 7

Conclusion

In this concluding chapter, the work in this thesis is summarized and the research achievements are listed, before possible future work is given.

7.1 Summary of the Thesis

Haptic visualization is a growing research area which is concerned with conveying the underlying data of a presentation using the sense of touch. Many challenges remain, whether at the application level or at the development level. This thesis has tackled both of these challenges and proposed some solutions for each.

In Part I, haptic data visualization has been examined at the application level. First, the area has been thoroughly reviewed (in Chapter 2) using a classification inspired from Bertin [Ber83] and Lohse et al. [LRBW90, LBWR94]. The categories – Charts, Maps, Signs, Networks, Diagrams, Images & Photo-realistic Renderings and Tables – organize the different techniques according to the data representation, rather than the data type or the display type. For each of these techniques, the different designs for the representation and interactions, the device used and the results of evaluation, when conducted, are detailed. Indeed, representation, interactions and hardware are the three main challenges faced by developers during the design of a haptic application. Their exploration across the categories and various types of hardware feedback emphasized common strategies and metaphors that are used. Moreover, the review also highlighted areas well investigated, where these common strategies can be found, learnt and extended to other, less well investigated, areas.

Even in areas that are well investigated, there are remaining challenges to convey data. For instance, in the case of line charts, difficulties in conveying a haptic overview, or in differentiating several lines persist. This is the reason why, in Chapter 3, navigation techniques, in particular guidance techniques, are explored to tackle the haptic overview challenge. A haptic line chart application was developed using the H3DAPI which aimed at conveying the general shape of the line as well as information about the different points of interest (i.e. extrema and intersections with axes). The representation of the line chart involved a V-shape and embossed axes, similar to previous work reviewed in Chapter 2. However, as opposed to previous work, the display spanned across both negative and positive values on the axes, and small bumps were added to the line shape to provide information about axis crossing. Three guidance interactions were implemented, inspired from ideas presented by Roberts et al. [RFC02] but which had not previously been implemented, that take the user along the line, with different degrees of constraint. The museum tour tightly constrains the user to the line, but at points of interest, the tour stops for a given time, before resuming, where the user is given a wider range of movement to explore the point surroundings. On the other hand, the water skier and bus tours are both continuous movements that slow down at points of interest to indicate their presence. The water skier tour allows the user to move side to side whereas the bus tour completely constrains the user to the path.

In other areas, such as maze navigation or handwriting, guidance has improved learning and it was hypothesized that guidance metaphors coupled with free exploration would contribute to building a better mental image of the chart, thus helping to provide an overview. Therefore, a pilot study was conducted with three participants. They were given a series of charts, with three different levels of complexity (easy, medium and challenging) and asked to explore them in three conditions (free exploration alone, guidance alone using the bus tour, and a combination of free exploration and guidance) and then at the end of the exploration, to draw the line shape as well as counting and identifying the points of interest. The participants achieved on average 80.5% of correct answers for the free exploration condition, 75% for the fully guided constraint and 97.2% in the case of the combined mode with 100% of correct answers for easy and challenging graphs. The number of subjects is too small to consider these results significant and thus to draw any conclusions about the hypothesis tested. However, they are encouraging as they seem to support that free exploration combined with guidance-models improved the understanding of the important graph features and they underline the need to perform further research in the area, including evaluation. Other than support the hypothesis, the pilot study revealed a few usability issues with the design of the line chart application: being able to control the speed, sharp edges and a clearer indication of the start and end line points. Solutions were proposed for the first two issues, but not evaluated.

Part I revealed challenges at the development level, in particular those encountered when developing the line chart application (see end of Chapter 3). Haptic visualization requires the investigation and design of effective representations and especially interactions. However, for the latter, solid programming background is required and the process to design an idea for an interaction, implement it and test it can be tedious and long. In Part II, solutions to similar problems in other domains were surveyed and a prototyping tool was developed.

Two holistic approaches to assisting the design of haptic interactions are surveyed in Chapter 4. These consist of providing taxonomies and frameworks to guide the design, implementation and evaluation of haptic applications and interactions, and providing prototyping languages to allow for rapid implementation and testing of haptic interactions. Many guidelines for haptic design are available and even being standardized through the 'ISO 9241-920 Ergonomics of human-system interaction – Guidance on tactile and haptic interactions' [vEK08] standard, which covers the issues related to tactile and haptic interactions. Although these guidelines are not specific to haptic visualization, they will include many applicable guidelines as well as more general and certainly useful information about haptic design. The methodologies and frameworks presented to guide the development are also not specific to the haptic visualization area, but their general process (design based on guidelines and iterative development) can surely be reused.

The other approach tackles rapid development by providing prototyping languages. These languages are either XML-based or visual, focusing on reusing and extending common interaction techniques or on device configurability where any device can be mapped to an interaction technique. Many different solutions have been proposed (see Chapter 4), however hardly any clearly support haptic, in particular force-feedback, devices. Moreover, most of them are aimed at programmers and involve some programming. A prototyping solution accessible to non-programmers, such as designers or teachers of visually impaired people, would surely be beneficial and promote the brainstorming of ideas: as haptic is an active sense, being able to get a feeling, even approximate, of the results of an idea, is likely to be as efficient as visual mock-ups for a visual website.

Therefore, the rest of the thesis investigated the development of a prototyping tool for haptic interactions aimed at non-programmers (see Chapters 5 and 6). The implemented tool, HITPROTO (Haptic Interaction Techniques PROtotyping TOol), is built on the separation of the representation and the interaction specification and helps prototyping the latter. As the representation influences the design of interactions, it can be loaded as an X3D scene into the tool and the scene objects can be acted upon within the tool. HITPROTO is based on a dataflow model, where the user assembles visual blocks into an interaction scenario, which can then be executed and tested. The blocks are divided into 'Action' blocks, which correspond to tasks and effects (e.g. highlight, remove, set guidance, start guidance) and 'Flow' blocks, which control the behaviour of the tasks (i.e. when and how tasks should happen, such as after a keyboard press or when the movement passes through a defined point of interest). The blocks included so far in the prototype cover guidance interactions (such as the ones presented in the line chart application), the addition of haptic effects and the haptic highlighting of objects (see Chapter 5 for the tool description and Chapter 6 for examples of prototyped visualization scenarios). HITPROTO was developed on top of the H3DAPI, a high-level haptic API that interfaces several force-feedback devices. Therefore, HITPROTO should be able to prototype interactions for any of the devices supported by the H3DAPI, but in practice the prototype was only tested with the desktop PHANTOM device. The H3DAPI was used to inform the choice of parameters for the blocks (some blocks correspond directly to elements from the API while other blocks are combinations of such elements) but is mostly used for haptic rendering, when the user is executing the scenario. After linking the blocks into a scenario, the diagram is parsed and Python code is automatically generated, which calls to the API. Python code has been chosen, rather than directly instantiating the diagram in C++, to provide the user with code that can be used to learn how to program with the API when the user wants to become more advanced or that can be extended into more complex interactions by a programmer.

As HITPROTO aims to facilitate the rapid prototyping of haptic interactions by non-programmers, it was evaluated by conducting a usability study, described in Chapter 6. This study was divided into: a familiarization with the setup phase, a tutorial introducing the tool and how to use it by walking the participants through examples, and the evaluation, where the participants were given four interaction scenarios descriptions and asked to prototype them until they decided the scenarios fulfilled the description. During the evaluation, participants were allowed to ask for help, although encouraged not to, and the amount of help provided was noted as well as measures of completion times and errors. A pilot study was first conducted to find an optimum design for the experiment. Nine postgraduate students were recruited with no programming skills, and they also had no or little experience with haptic technologies. The results obtained were quite positive; overall, participants could use HITPROTO to prototype haptic interactions: 88.9% of the attempts at the tasks resulted in a working interaction, with or without help, while only 8.3% of them resulted in failures, despite the help given. Many of the participants needed some help; but four participants commented that this was due to the lack of time to learn how to use the tool and familiarize themselves with it. In fact, eight participants out of nine found the tool easy to use, especially after some practice. The evaluation also underlined areas for improvement and in particular, several of the participants asked for a tooltip functionality, which has been added after the experiment.

Therefore, people with no or little programming skills could use HITPROTO and understand the logic behind the dataflow language to program haptic interactions. Furthermore, they managed to do so in a relatively short amount of time with a short training period given the novelty of the tool to them. This reinforces the general hypothesis that learning how to operate such a tool would take less time and be more beneficial than learning how to use the haptic API and the corresponding programming languages (such as C++); thus promoting the development of haptic interactions, particularly for haptic data visualization.

7.2 Contributions

By exploring the design space of haptic interactions in the context of data visualization, tackling challenges both at the application level and at the development level, several contributions have been made. They are listed and summarized below.

7.2.1 Comprehensive Review of Designs for Haptic Data Visualization

As explained in the paper “Where are We with Haptic Visualization” [RP07], haptic visualization is a recent area. Although a few reviews, classified according to application area, are available, no reviews was found which surveyed the techniques for tackling the challenges of haptic visualization. However, reviews of existing work are useful and necessary, especially when an area is not well established, as they help collate the different designs and ideas, draw conclusions about the research conducted so far, identify gaps or unsolved challenges and draw inspiration for future work.

Therefore, the first contribution is a comprehensive review of the designs for haptic visualization, which categorized them by the type of representation (Chart, Map, Signs, Networks, Diagrams, Images & Photo-realistic Renderings and Tables) rather than the application area or data type. Each category describes the existing techniques with details about the representation and interactions metaphors, the hardware used and the results of evaluation if any was conducted. In addition to providing a comprehensive list of achievements, the review also highlights areas well investigated, and their remaining challenges, and areas poorly investigated, which gives directions for future work. This review was also published in the article entitled “Review of Designs for Haptic Data Visualization” [PR09] which will be published in the IEEE journal ‘Transactions on Haptics’.

7.2.2 New Designs for Haptic Charts

The second contribution is at the application level and provides designs to solve the challenge of the haptic overview in the case of haptic charts. Even though haptic line charts have been quite well investigated, many challenges remain, including the haptic overview which has not been much investigated in that domain, whereas haptic scatter plots have hardly been researched.

Therefore, the contribution is two-fold: guidance metaphors have been implemented and tested with a pilot study to convey the haptic overview of a line chart (described in Chapter 3); while the overview of a scatter plot is conveyed with a force depending on the point concentration around the device through the sum of the inverse of distances from each point to the device (see Chapter 6). The guidance metaphors were also presented at the Haptic and Audio Interaction Design (HAID) conference, as a poster [PR07] and later in a paper [PRR09]. The details of the method for haptic scatter plot overview were also published in that paper [PRR09].

7.2.3 Haptic Interaction Techniques Prototyping Tool

The last contribution answered the need for prototyping tools to facilitate the design of haptic interactions and make them accessible to a wider audience, including designers. The contribution includes the development and implementation of a visual prototyping tool for haptic interactions (HITPROTO) as well as its evaluation to assess whether people with no programming knowledge could effectively use the tool (see chapters 5 and 6). HITPROTO is based on a dataflow model which uses visual blocks to prototype haptic interactions. Although still a prototype, HITPROTO demonstrated the prototyping of interactions through a series of case studies and through an in-depth qualitative evaluation with nine participants, which provided good results overall. The design, implementation and testing of HITPROTO has been accepted for the Haptics Symposium proceedings [PRR10].

7.3 Future Work

As haptic visualization is a relatively infant area, there is still much work to be done. This thesis has tackled some of the challenges, but many still remain and need to be addressed. These challenges cover several directions: (1) extending and improving the current designs, (2) investigating other techniques and areas, related to the current designs and not covered in this work, and (3) following new lines of research.

7.3.1 Extending the Current Designs

Haptic Line Chart

The pilot study exhibited promising results, however to validate the usefulness of guidance interaction techniques to convey the haptic overview of a line chart, a quantitative study with at least 15 participants should be conducted in order to obtain statistically significant results, based on the pilot study methodology. A quantitative study could prove the hypothesis that the combined free exploration and guidance models with different levels of constraint can provide the chart overview and improve the understanding of the important chart features.

As for the design of the guidance interaction techniques, more behaviour at the points of interest should be investigated. Ideas for such behaviour include adding magnetic lines leading to the axes to provide an estimate of the coordinate values as well as allowing the user to leave haptic landmarks to easily find the points during free exploration. The guidance techniques could also be extended to other types of charts, such as bar charts, where the path could take the user along each bar and back to the x-axis or along the envelope of the bar chart.

It is also envisaged that the guidance exploration techniques would be useful for distinguishing multiple lines. Indeed, with a guidance that would take the user along each line and give information about intersections with other lines, the user could get a general idea of the lines shapes, their relative locations to each other as well as their intersections, which could be then confirmed during free exploration. As distinguishing multiple lines is still an unsolved challenge, the guidance metaphors should also be evaluated for this challenge.

Haptic Scatter Plot

Similar to the guidance metaphors presented for the line chart application, the haptic overview technique described for scatter plots needs to be evaluated to demonstrate that information is conveyed effectively and to test whether the technique succeeds in providing a general overview of the plot trend. An experiment presenting several scatter plots with different correlations and with the task of identifying the correlations could be conducted. The experiment could also include several datasets and test whether users can discriminate the different datasets using the overview scenario and whether they can also identify the correct correlation for each.

Furthermore, the force model for the scatter plot interaction technique should be perfected, as the one presented lack accuracy for the point density and new models should be explored and tested, for instance using the inverses of the squared distances as commonly used in physics.

Additionally, the interaction technique presented does not provide information about the absolute position of the datasets, relative to axes. This could be achieved using some guidance interactions, which similarly to the line application, could take the user along the axes before taking them to the beginning of the plot.

Also, the force model for the overview could be complemented with a haptic representation of the scatter plot. Such a representation has already been proposed by Crossan et al. [CWMS04]. The combination of this existing representation with the force model could be studied or perhaps a new representation should be investigated.

HITPROTO

The evaluation of HITPROTO highlighted areas for improvement of the current design. These include incorporating an error-checking and compilation mechanism (sometimes some blocks were not well connected or some end blocks missing and the scenario would execute but was not semantically correct), changing some terms that may be too technical, providing a snap-to-grid or aligning blocks functionality and integrating the manual into a help section, to supplement the tooltip functionality that has already been added. The usability study gave valuable insights into the ease of use of the tool and the improvements that needed to be made. Nonetheless, a quantitative study should also be conducted. Indeed, an experiment evaluating the benefits of the tool as compared to some benchmarks, to other similar tools or to learning and programming with the API directly would validate the usefulness of the tool and the hypothesis that such a tool would speed up the prototyping process.

More generally, the number of interaction techniques currently available in the HITPROTO prototype should be extended. As explained by De Boeck et al. [DBRC05], only a limited set of existing interaction techniques can be transferred to the haptic domain, and as underlined by De Felice et al. [DFAD09], this set is even more restricted in the case of haptic visualization applications. Initially, the interaction techniques presented by De Felice et al. [DFAD09] should be included in HITPROTO such as 'zooming', to cover all the haptic visualization interaction techniques. In 'non-visual' visualization (usually haptic and audio modalities), approaches combining haptics and audio have proven to be more efficient than using either modality alone. Therefore, in a second stage, audio effects could be integrated in the tool. Finally, the interactions transferable to the haptic domain, as summarized by De Boeck et al. [DBRC05], could be included to complete the tool so that it is not restricted to the non-visual visualization domain and therefore extended to include all haptic interactions.

In theory HITPROTO should work with multiple force feedback devices, however in practice it was only tested with the PHANTOM device. Hence, testing with other devices, such as the Novint Falcon, would demonstrate that it is widely applicable.

Finally, general issues of encapsulation and extensibility and their possible integration should also be tackled to offer a complete tool. Encapsulation would allow saving a set of blocks, with their parameters, into a custom block, which could be then reused directly into other scenarios. For instance, the museum tour could be saved as a 'museum tour' block, which could be reused in a scenario that also allows highlighting objects haptically and so would test the combination of the two interactions. This would avoid clutter of the blocks on the diagram. Other functionalities such as zooming or a graphical visualization similar to the fisheye view could also be used for dense diagrams for which the display can become easily cluttered. Extensibility, on the other hand, would permit the user/developer to add new blocks to HITPROTO, for instance, by providing the block interface and generated Python code.

7.3.2 Investigating New Techniques or Areas

Haptic Interaction Techniques

This thesis presented a few interaction techniques for haptic data visualization as well as ideas to extend them. Certainly more interaction techniques should be researched and evaluated in the context of haptic data visualization. For instance, the areas which lack haptic visualization solutions as highlighted by the review in Chapter 2 could be investigated. These include *networks* and *tables*, where a force model, equivalent in concept to the overview scatter plot technique, could be tried, with the force depending on the value of each table cell and thus helping identify patterns.

In general, more interaction techniques should be explored, covering not only the issues of context, but also navigation and qualitative and quantitative information. They should be designed to help achieve the principal tasks necessary in visualization as stated by Ben Schneiderman's mantra "Overview first, zoom and filter, then details on demand". In particular, the challenge of displaying value haptically, e.g. a haptic legend or the axis numbers, has hardly been tackled, primarily due to the perceptual limitations of the haptic sense.

HITPROTO

HITPROTO aims to help the user rapidly prototype haptic interactions, in particular for force-feedback devices. Nevertheless, the tool could be augmented with the prototyping of tactile interactions. Prototyping tools for vibrotactile icons already exist and could be merged into the tool. In particular, the work by Lee et al. [LRC09], who use a musical metaphor, is particularly interesting as they share the same goal of accessibility to non-technical users.

By integrating tactile feedback, the tool could in turn expand its areas of application from desktop applications to mobile phone applications or other types of display. Indeed, in the last years, mobile applications have received considerable interest in the haptic community where new ways of conveying the information unobtrusively and of compensating for the small display size are being researched. Furthermore, the tool could be made freely available and open source to promote its development.

7.3.3 Following New Research Leads

Lastly, this thesis has explored the development of interaction techniques and encountered issues related to representation, interactions and hardware limitations. Two main approaches that can facilitate and thus enhance the prototyping process have been identified and one of these approaches, prototyping languages or tools, has been examined and led to the implementation of HITPROTO.

The second approach concerns taxonomies and frameworks to guide the design and implementation of applications. As described in Chapter 4, the taxonomies and guidelines have been well investigated for haptic applications, and are being standardized. However, a framework for aiding the design of interactions (and applications) for haptic visualization has not yet been found. Nesbitt [Nes03] proposed a framework for multisensory applications, but it does not focus on 'non-visual' visualization (i.e. all senses but vision, and most commonly audio and haptics), where the lack of vision needs to be compensated for. Moreover, in terms of interactions, a framework based on a taxonomy by interaction type, as presented by Bowman and Hodges [BH99], seems more intuitive. Therefore, developing a framework for the development of haptic interactions in visualization, which would combine ideas both from Bowman and Hodges [BH99] and from Nesbitt [Nes03], is a direction that might also be pursued. As part of that framework, guidelines specific to haptic visualization, based on existing research, could be compiled and further work could attempt to derive more guidelines.

Bibliography

[3ds10] 3ds Max, a 3D modelling, animation and rendering software. `http://www.autodesk.co.uk/3dsmax` [Accessed: 03-01-2010], 2010.

[AHDBL09] C. Appert, S. Huot, P. Dragicevic, and M. Beaudouin-Lafon. Flow-States : prototypage d'application interactives avec des flots de données et des machines à états. In *International Conference on Association Francophone d'Interaction Homme-Machine (IHM '09)*, pages 119–128, Grenoble, France, October 2009. ACM Press.

[AS96] Ricardo S. Avila and Lisa M. Sobierajski. A Haptic Interaction Method for Volume Visualization. In Roni Yagel and Gregory M. Nielson, editors, *Conference on Visualization (VIS '96)*, pages 197–204, San Francisco, California, USA, 1996. IEEE Computer Society Press.

[BB04] S. A. Brewster and L. M. Brown. Tactons: Structured Tactile Messages for Non-Visual Information Display. In *Australasian User Interface Conference (AUIC'04)*, pages 15–23, Dunedin, New Zealand, January 2004. Australian Computer Society.

[BB07] Christoph W. Borst and Vijay B. Baiyya. Enhancing VR-based Visualization with a 2D Vibrotactile Array. In *Symposium on Virtual Reality Software and Technology (VRST '07)*, pages 183–186, Newport Beach, California, 2007. ACM press.

[BBP05] Lorna M. Brown, Stephen A. Brewster, and Helen C. Purchase. A First Investigation into the Effectiveness of Tactons. In *World-Haptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05)*, pages 167–176, Pisa, Italy, 2005. IEEE Computer Society Press.

[BDK$^+$04] Cagatay Basdogan, Suvranu De, Jung Kim, Manivannan Muniyandi, Hyun Kim, and Mandayam A. Srinivasan. Haptics in Minimally Invasive Surgical Simulation and Training. *IEEE Computer Graphics and Applications*, 24(2):56–64, March-April 2004.

[BDW$^+$03] Aaron Bloomfield, Yu Deng, Jeff Wampler, Pascale Rondot, Dina Harth, Mary McManus, and Norman Badler. A taxonomy and comparison of haptic actions for disassembly tasks. In *IEEE Virtual Reality (VR'03)*, pages 225–231, Los Angeles, CA, USA, March 2003. IEEE Computer Society Press.

[Ber83] Jacques Bertin. *Semiology of Graphics.* The University of Wisconsin Press, 1983. Translated by William J. Berg.

[BGF$^+$03] S. Bosman, B. Groenendaal, J. W. Findlater, T. Visser, M. de Graaf, and P. Markopoulos. GentleGuide: An Exploration of Haptic Output for Indoors Pedestrian Guidance. In *International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'03)*, volume 2795 of *Lecture Notes in Computer Science*, pages 358–362, Udine, Italy, 2003. Springer.

[BH99] Doug A. Bowman and Larry F. Hodges. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. *Journal of Visual Languages & Computing*, 10(1):37–53, February 1999.

[BJOYBK90] Frederick P. Brooks Jr., Ming Ouh-Young, James J. Batter, and P. Jerome Kilpatrick. Project GROPE - Haptic Displays for Scientific Visualization. In *Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH'90)*, pages 177–185. ACM Press, 1990.

[BK05] Stephen Brewster and Alison King. An Investigation into the Use of Tactons to Present Progress Information. In Maria Francesca Costabile and Fabio Patern, editors, *IFIP Conference on Human-Computer interaction - INTERACT'05*, pages 6–17, Rome, Italy, 2005. Springer.

[BKH97] D. A. Bowman, D. Koller, and L. F. Hodges. Travel In Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control

Techniques. In *Virtual Reality Annual International Symposium (VRAIS '97)*, pages 45–52, Albuquerque, New Mexico, March 1997. IEEE Computer Society Press.

[BKHAK04] Mohamed Benali-Khoudja, Moustapha Hafez, Jean-Marc Alexandre, and Abderrahmane Kheddar. Tactile interfaces: a state-of-the-art survey. In *International Symposium on Robotics ISR'04*, pages 721–726, Paris, France, March 2004.

[BL93] G. C. Burdea and N. A. Langrana. Virtual Force Feedback: Lessons, Challenges, Future Applications. *Journal of Robotics and Mechatronics*, 5(2):178–182, 1993. Also appeared as Invited paper, Proceedings of 1992 A.S.M.E. Winter Annual Meeting - Advances in Robotics, DSC-Vol. 42, Anaheim CA, pp. 41-47, 1992.

[Ble10] Blender, "a free open source 3D content creation suite". `http://www.blender.org/` [Accessed: 03-01-2010], 2010.

[BMF03] Tim Beamish, Karon MacLean, and Sidney Fels. Designing the Haptic Turntable for Musical Control. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, page 24, Los Angeles, CA, USA, March 2003. IEEE Computer Society Press.

[BMF04] Timothy Beamish, Karon Maclean, and Sidney Fels. Manipulating Music: Multimodal Interaction for DJs. In *Conference on Human Factors in Computing Systems (CHI'04)*, pages 327–334, Vienna, Austria, April 2004. ACM Press.

[BMS01] Stephen Brewster and Roderick Murray-Smith, editors. *Haptic Human-Computer Interaction First International Workshop (2000)*, volume 2058 of *Lecture Notes in Computer Science*. Springer, August 2001.

[BNB04] Jullien Bouchet, Laurence Nigay, and Didier Balzagette. ICARE: a component-based approach for multimodal interaction. In *French-speaking conference on Mobility and ubiquity computing (UbiMob '04)*, pages 36–43, Nice, France, June 2004. ACM Press.

[BO06] Kevin Bowen and Marcia K. O'Malley. Adaptation of Haptic Interfaces for a LabVIEW-based System Dynamics Course. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'06)*, pages 147–152, Alexandria, Virginia, USA, March 2006. IEEE Computer Society Press.

[Boo08] Wayne Book. Control Prototyping with LabVIEW for Haptic and Telerobotic Systems. Webcast, March 2008. Available at: `http://zone.ni.com/wv/app/doc/p/id/wv-19` [Accessed: 01-01-10].

[BPK05] B. Bayart, A. Pocheville, and A. Kheddar. An adaptive haptic guidance plug-in for I-TOUCH: example through handwriting teaching simulation and 3D maze. In *International Workshop on Haptic Audio Visual Environments and their Applications (HAVE'05)*, pages 6 pp.–, Ottawa, Ontario, Canada, October 2005. IEEE Computer Society Press.

[Bro92] K. W. Brodlie. Visualization techniques. In K.W. Brodlie, L.A. Carpenter, R.A. Earnshaw, J.R. Gallop, R.J. Hubbold, A.M. Mumford, C.D. Osland, and P. Quarendon, editors, *Scientific Visualization: Techniques and Applications*, chapter 3, page 37. Springer, 1992.

[Bro96] John Brooke. SUS - A quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, and A. L. McClelland, editors, *Usability Evaluation in Industry*, pages 189–194. Taylor & Francis, 1996.

[BRS+92] G. Burdea, E. Roskos, D. Silver, F. Thibaud, and R. Wolpov. A Distributed Virtual Environment with Dextrous Force Feedback. In *Interface to Real and Virtual Worlds Conference*, pages 255–265, Montpellier, France, 1992.

[BRSB03] Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments. In *Conference on Human Factors in Computing Systems (CHI '03)*, pages 537–544, Ft. Lauderdale, Florida, USA, 2003. ACM Press.

[BS02a] C. Basdogan and M. A. Srinivasan. Haptic Rendering in Virtual Environments. In Kay M. Stanney, editor, *Handbook of Virtual Environments: Design, Implementation, and Applications*, chapter 6, pages 117–134. Lawrence Erlbaum Associates, 2002.

[BS02b] S. James Biggs and Mandayam A. Srinivasan. Haptic Interfaces. In K. M. Stanney, editor, *Handbook of Virtual Environments: Design, Implementation, and Applications*, chapter 5, pages 93–115. Lawrence Erlbaum Associates, 2002.

[BT07] Hans V. Bjelland and Kristian Tangeland. User-Centered Design Proposals for Prototyping Haptic User Interfaces. In I. Oakley and S. Brewster, editors, *International Workshop on Haptic and Audio Interaction Design (HAID'07)*, volume 4813 of *Lecture Notes in Computer Science*, pages 110–120, Seoul, Korea, November 2007. Springer.

[BZR$^+$92] G. Burdea, J. Zhuang, E. Roskos, D. Silver, and N. Langrana. A Portable Dextrous Master with Force Feedback. *Presence*, 1(1):18–28, 1992.

[Cas05] Géry Casiez. *Interface Homme-Machine: Le retour haptique.* Polytech-Lille / Telecom Lille I, October 2005.

[CE00] B. P. Challis and A. D. N. Edwards. Weasel: a computer based system for providing non-visual access to music notation. *ACM SIGCAPH Computers and the Physically Handicapped*, (66):1–12, January 2000.

[CFF$^+$05] Jim Carter, David Fourney, Shinichi Fukuzumi, John A. Gardner, Yasuo Horiuchi, Gunnar Jansson, Helmut Jurgensen, Roland Kadefors, Tadashi Kobayashi, Misa Grace Kwok, Manabi Miyagi, and Keith V. Nesbitt. The GOTHI model of tactile and haptic interaction. In *Guidelines On Tactile and Haptic Interactions (GOTHI'05)*, pages 93–96, Saskatoon, Saskatchewan, CANADA, October 2005. Proceedings available online at: `http://www.cs.usask.ca/research/research_groups/userlab/GOTHI/GOTHI-05%20Proceedings.html` [Accessed: 01-01-10].

[Cha00] Ben Challis. *Design principles for tactile communication within the human-computer interface.* PhD thesis, University of York, September 2000.

[Cha02] Daniel Chandler. *Semiotics: the basics.* Routledge, 2002.

[Chu02] Lonny L. Chu. Haptic Design for Digital Audio. In *International Conference on Multimedia and Expo (ICME'02)*, volume 2, pages 441–444, Lausanne, Switzerland, 2002. IEEE Computer Society Press.

[CK07] Mathis Csisinko and Hannes Kaufmann. Towards a Universal Implementation of 3D User Interaction Techniques. In *IEEE Virtual Reality – Workshop on "Specification, Authoring, Adaptation of Mixed Reality User Interfaces" (MRUI'07)*, pages 17–24, Charlotte NC, USA, March 2007. IEEE Computer Society Press.

[CM97] S. K. Card and J. Mackinlay. The Structure of the Information Visualization Design Space. In *Symposium on Information Visualization (InfoVis '97)*, pages 92–99, Phoenix, AZ, USA, Octobre 1997. IEEE Computer Society Press.

[CMH05a] Vasilios G. Chouvardas, Amalia N. Miliou, and Miltiadis K. Hatalis. Tactile display applications: A state of the art survey. In *Balkan Conference in Informatics (BCI'05)*, pages 290–303, Ohrid, FYROM, 2005.

[CMH05b] Vasilios G. Chouvardas, Amalia N. Miliou, and Miltiadis K. Hatalis. Tactile Displays: a short overview and recent developments. In *International Conference on Technology and Automation (ICTA'05)*, pages 246–251, Thessaloniki, Greece, October 2005.

[CMM04] Andrew Chan, Karon MacLean, and Joanna McGrenere. Learning and Identifying Haptic Icons under Workload. Technical Report TR-2004-15, Sensory perception and interaction research group, 2004.

[CPK$^+$98] C. Colwell, H. Petrie, D.E. Kornbrot, A. Hardwick, and S. Furner. Use of a haptic device by blind and sighted people: perception of virtual textures and objects. In I.B.Placencia and E. Porrero, editors, *Improving the Quality of Life for the European Citizen: Technology for Inclusive Design and Equality.* IOS Press, 1998.

[CT97] M. K. D. Coomans and H. J. P Timmermans. Towards a Taxonomy of Virtual Reality User Interfaces. In *Information Visualization (IV'97)*, pages 279–284. IEEE Computer Society Press, 1997.

[CWMS04] Andrew Crossan, John Williamson, and Roderick Murray-Smith. Haptic Granular Synthesis: Targeting, Visualisation and Texturing. In *Information Visualisation (IV'04)*, pages 527–532, Maynooth, Ireland, July 2004. IEEE Computer Society Press.

[Dat10] Data and Story Library (DASL). Fisher's Irises. `http://lib.stat.cmu.edu/DASL/Stories/Fisher'sIrises.html` [Accessed: 03-01-2010], 2010.

[DBJ$^+$01] Kelly Dobson, Danah Boyd, Wendy Ju, Judith Donath, and Hiroshi Ishii. Creating Visceral Personal and Social Interactions in Mediated Spaces. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '01)*, pages 151–152, Seattle, Washington, 2001. ACM Press.

[DBRC05] Joan De Boeck, Chris Raymaekers, and Karin Coninx. Are Existing Metaphors in Virtual Environments Suitable for Haptic Interaction. In *Virual Reality International Conference (VRIC 2005)*, pages 261–268, Laval, France, April 2005.

[DBVRC07] Joan De Boeck, Davy Vanacken, Chris Raymaekers, and Karin Coninx. High-Level Modeling of Multimodal Interaction Techniques Using NiMMiT. *Journal of Virtual Reality and Broadcasting*, 4(2), September 2007. Available online at: `http://www.jvrb.org/4.2007/1161/` [Accessed: 02-01-2010].

[DF04] Pierre Dragicevic and Jean-Daniel Fekete. Support for Input Adaptability in the ICON Toolkit. In *International Conference on Multimodal Interfaces (ICMI '04)*, pages 212–219, State College, Pennsylvania, USA, October 2004. ACM Press.

[DFAD09] Fabio De Felice, Giovanni Attolico, and Arcangelo Distante. Configurable Design of Multimodal Non Visual Interfaces for 3D VE's.

In M. Ercan Altinsoy, Ute Jekosch, and Stephen Brewster, editors, *Haptic and Audio Interaction Design (HAID'09)*, volume 5763 of *Lecture Notes in Computer Science*, pages 71–80, Dresden, Germany, September 2009. Springer.

[DMW+98] L. J. K. Durbeck, N. J. Macias, D. M. Weinstein, C. R. Johnson, and J. M. Hollerbach. SCIRun Haptic Display for Scientific Visualization. In *Third Phantom Users Group Workshop (PUG'98)*, MIT RLE Report TR624, Dedham, MA, 1998. MIT Press.

[EAAE08] Mohamad Eid, Sheldon Andrews, Atif Alamri, and Abdulmotaleb El Saddik. HAMLAT: A HAML-Based Authoring Tool for Haptic Application Development. In M. Ferre, editor, *Eurohaptics (EH'08)*, volume 5024 of *Lecture Notes in Computer Science*, pages 857–866, Madrid, Spain, June 2008. Springer.

[EAS06] Mohamad Eid, Atif Alamri, and Abdulmotaleb El Saddik. MPEG-7 Description of Haptic Applications Using HAML. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE '06)*, pages 134–139, Ottawa, Canada, November 2006. IEEE Computer Society Press.

[ELW+98] S. Ertan, C. Lee, A. Willets, H. Tan, and A. Pentland. A Wearable Haptic Navigation Guidance System. In *International Symposium on Wearable Computers (ISWC '98)*, pages 164–165, Pittsburgh, PA, USA, 1998. IEEE Computer Society Press.

[EM03] Mario J. Enriquez and Karon E. MacLean. The Hapticon Editor: A Tool in Support of Haptic Communication Research. In *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS'03)*, pages 356–362, Los Angeles, CA, USA, March 2003. IEEE Computer Society Press.

[EMC06] Mario Enriquez, Karon MacLean, and Christian Chita. Haptic Phonemes: Basic Building Blocks of Haptic Communication. In *International Conference on Multimodal Interfaces (ICMI '06)*, pages 302–309, Banff, Alberta, Canada, 2006. ACM Press.

[Fau03] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383., August 2003.

[FB96] Jason P. Fritz and Kenneth E. Barner. Design of a Haptic Graphing System. In Anthony Langton, editor, *Rehabilitation Engineering and Assistive Technology Society of North America Conference (RESNA'96)*, pages 158–160, Salt Lake City, UT, USA, June 1996. RESNA.

[FB99] Jason P. Fritz and Kenneth E. Barner. Design of a Haptic Data Visualization System for People with Visual Impairments. *IEEE Transactions on Rehabilitation Engineering*, 7(3):372–384, September 1999.

[FD06] Z. Ben Fredj and D. A. Duce. GraSSML: Accessible Smart Schematic Diagrams for All. In *International Cross-Disciplinary Workshop on Web Accessibility (W4A'06)*, pages 57–60, Edinburgh, U.K., May 2006. ACM Press.

[FFM$^+$04] Brian Fisher, Sidney Fels, Karon MacLean, Tamara Munzner, and Ronald Rensink. Seeing, hearing, and touching: putting it all together. In *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04) Course Notes*, Los Angeles, CA, USA, 2004. ACM Press.

[FGH02] Pablo Figueroa, Mark Green, and H. James Hoover. InTml: a Description Language for VR Applications. In *3D Web technology (Web3D '02)*, pages 53–58, Tempe, Arizona, USA, February 2002. ACM Press.

[Fra07] Keith Franklin. *Non-Visual Data Visualization: towards a better design.* PhD thesis, Computer Science, University of Kent, 2007.

[FW06] N. Forrest and S.A. Wall. Protohaptic: Facilitating Rapid Interactive Prototyping of Haptic Environments. In David McGookin and Stephen Brewster, editors, *Workshop on Haptic and Audio Interaction Design (HAID'06)*, volume 4129 of *Lecture Notes in Computer Science*, pages 21–25, Glasgow, UK, August–September 2006. Springer.

[GP96] A. Gardiner and C. Perkins. Feel the bunkers: Tactile maps for blind golfers. In *Maps and Diagrams for Blind and Visually Impaired People: Needs, Solutions, Developments*, pages 26–36, Ljubljana, Slovenia, October 1996. ICA Commission on Maps and Graphics for Blind and Visually- Impaired People.

[Gum06] Simone Gumtau. Freshly squeezed touch into bits: towards the development of a haptic design palette. *Virtual Reality*, 9(4):250–259, April 2006.

[H3D10] H3DAPI, by SenseGraphics AB. `http://www.h3dapi.org/` [Accessed: 03-01-2010], 2010.

[HAB07] Eve Hoggan, Sohail Anwar, and Stephen A. Brewster. Mobile Multi-Actuator Tactile Displays. In Ian Oakley and Stephen Brewster, editors, *Haptic and Audio Interaction Design Workshop (HAID'07)*, number 4813 in Lecture Notes in Computer Science, pages 22–33, Seoul, Korea, 2007. Springer.

[HACH$^+$04] Vincent Hayward, Oliver R. Astley, Manuel Cruz-Hernandez, Danny Grant, and Gabriel Robles-De-La-Torre. Haptic Interfaces and Devices. *Sensor Review*, 24(1):16–29, February 2004.

[Hap04] Haptics Photo Gallery. `http://haptic.mech.northwestern.edu/intro/gallery/index.html` [Accessed: 03-01-2010], 2004.

[Har09] W.S. Harwin. Introduction to haptics and haptic technologies. Available online at: `http://www.personal.rdg.ac.uk/~shshawin/LN/L8hapticdesigns.html` [Accessed: 02-01-2010], April 2009.

[HB07] Eve Hoggan and Stephen Brewster. New Parameters for Tacton Design. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '07)*, pages 2417–2422, San Jose, CA, USA, 2007. ACM Press.

[HBJ08] Eve Hoggan, Stephen A. Brewster, and Jody Johnston. Investigating the Effectiveness of Tactile Feedback for Mobile Touchscreens. In *Conference on Human Factors in Computing Systems (CHI '08)*, pages 1573–1582, Florence, Italy, 2008. ACM Press.

[HDD$^+$04] Stéphane Huot, Cédric Dumas, Pierre Dragicevic, Jean-Daniel Fekete, and Gérard Hégron. The MaggLite Post-WIMP Toolkit: Draw It, Connect It and Run It. In *User Interface Software and Technology (UIST '04)* [JDM99], pages 257–266.

[HFWW04] B. S. Hoyle, J. M Fowler, D. A. Waters, and D. J. Withington. Development of the electronic guide cane for enhanced primary mobility for the vision impaired. In *Conference and Workshop on Assistive Technologies for Vision and Hearing Impariment: State-of-the-Art and New Challenges (CVHI'04)*, Granada, Spain, 2004. Available online at: `http://forte.fh-hagenberg.at/Project-Homepages/Blindenhund/conferences/granada/papers/HOYLE/hoyle.htm` [Accessed: 02-01-2010].

[HHK$^+$04] M. Horstmann, Cornelius Hagen, A. King, S. Dijkstra, D. Crombie, G. Evans, G. Ioannidis, P. Blenkhorn, O. Herzog, and Christoph Schlieder. TeDUB: Automatic Interpretation and Presentation of Technical Diagrams for Blind People. In *Conference and Workshop on Assistive Technologies for Vision and Hearing Impariment: State-of-the-Art and New Challenges (CVHI'04)*, Granada, Spain, 2004. Available online at: `http://forte.fh-hagenberg.at/Project-Homepages/Blindenhund/conferences/granada/papers/HORSTMANN/horstmann.html` [Accessed: 02-01-2010].

[HK05] S. J. Haase and K. A. Kaczmarek. Electrotactile perception of scatterplots on the fingertips and abdomen. *Medical and Biological Engineering and Computing*, 43(2):283–289, April 2005.

[HM90] R. B. Haber and D. A. McNabb. Visualization idioms: a conceptual model for scientific visualization systems. In G. M. Nielson, B. Shriver, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, 1990.

[HMB03] Zayd Hendricks, Gary Marsden, and Edwin Blake. A MetaAuthoring Tool for Specifying Interactions in Virtual Reality Environments. In *Conference on Computer graphics, Virtual Reality,* *Visualisation and Interaction in Africa (AFRIGRAPH '03)*, pages 171–180, Cape Town, South Africa, 2003. ACM Press.

[HS04] Kelly S. Hale and Kay M. Stanney. Deriving Haptic Design Guidelines from Human Physiological, Psychophysical, and Neurological Foundations. *IEEE Computer Graphics Applications*, 24(2):33–39, 2004.

[IBL+99] F. Infed, S. Brown, C. Lee, L. Lawrence, A. Dougherty, and L. Pao. Combined Visual /Haptic Rendering Modes for Scientific Visualization. In *ASME International Congress on Dynamic Systems and Control Division*, volume DSC-Vol. 67, pages 93–99, Nashville, TN, November 1999. American Society of Mechanical Engineers (ASME).

[Imm10] Immersion Corporation. Immersion Studio. `http://www.immersion.com/products/rumble-technology/immersion-studio.html` [Accessed: 03-01-2010], 2010.

[IUS01] the NIST Industry USability Reporting project IUSR. Common Industry Format for Usability Test Reports. Technical report, National Institute of Standards and Technology, May 2001.

[Iwa90] H. Iwata. Artificial Reality with Force-feedback: Development of Desktop Virtual Space with Compact Master Manipulator. *Computer Graphics*, 24(4):165–170, 1990. Also published in the proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '90).

[JDM99] Robert J. K. Jacob, Leonidas Deligiannidis, and Stephen Morrison. A Software Model and SpecificationLanguage for Non-WIMP User Interfaces. *ACM Transactions on Computer-Human Interaction*, 6(1):1–46, March 1999.

[Jeo05] Wooseob Jeong. Multimodal Trivariate Thematic Maps with Auditory and Haptic Display. In Andrew Grove, editor, *The American Society for Information Science and Technology (ASIST '05)*, volume 42, North Carolina, USA, October-November 2005. ASIS&T Digital Library.

[JG02] Wooseob Jeong and Myke Gluck. Multimodal Bivariate Thematic Maps with Auditory and Haptic Display. In *International Conference on Auditory Display (ICAD'02)*, Kyoto, Japan, July 2002. Advanced Telecommunications Research Institute (ATR). Available online at: `http://www.icad.org/websiteV2.0/Conferences/ICAD2002/proceedings/66_WooseobJeong.pdf` [Accessed: 03-01-2010].

[JG03] Wooseob Jeong and Myke Gluck. Multimodal Geographic Information Systems: Adding Haptic and Auditory Display. *Journal of the American Society for Information Science and Technology (JASIST)*, 54(3):229–242, February 2003.

[JLP06] Lynette A. Jones, Brett Lockyer, and Erin Piateski. Tactile display and vibrotactile pattern recognition on the torso. *Advanced Robotics*, 20(12):1359–1374, 2006.

[JMRU06] Sandra Jehoel, Don McCallum, Jonathan Rowell, and Simon Ungar. An empirical approach on the design of tactile maps and diagrams: The cognitive tactualization approach. *British Journal of Visual Impairment*, 24(2):67–75, 2006.

[JP05] G. Jansson and P. Pedersen. Obtaining geographical information from a virtual map with a haptic mouse. In *International Cartographic Conference - "Maps for Blind and Visually Impaired" (ICC'05)*, La Coruna, Spain, July 2005. The International Cartographic Association (ICA-ACI).

[JSHG08] Caroline Jay, Robert Stevens, Roger Hubbold, and Mashhuda Glencross. Using Haptic Cues to Aid Nonvisual Structure Recognition. *ACM Transactions on Applied Perception*, 5(2):1–14, May 2008.

[JWVG08] Chris Jansen, Antoon Wennemers, Wouter Vos, and Eric Groen. FlyTact: A Tactile Display Improves a Helicopter Pilot's Landing Performance in Degraded Visual Environments. In Manuel Ferre, editor, *Eurohaptics (EH'08)*, volume 5024 of *Lecture Notes in Computer Science*, pages 867–875. Springer, 2008.

[KB05a] J. Kildal and S. A. Brewster. Explore the matrix: Browsing numerical data tables using sound. In Eoin Brazil, editor, *International Conference on Auditory Display (ICAD'05)*, pages 300–303, Limerick, Ireland, July 2005. Available online at: `http://www.icad.org/node/2552` [Accessed: 03-01-2010].

[KB05b] Stephen E. Krufka and Kenneth E. Barner. Automatic Production of Tactile Graphics from Scalable Vector Graphics. In *ACM SIGACCESS Conference on Computers and Accessibility (Assets '05)*, pages 166–172, Baltimore, MD, USA, 2005. ACM Press.

[KB07] Johan Kildal and Stephen A. Brewster. EMA-Tactons: Vibrotactile External Memory Aids in an Auditory Display. In *IFIP Conference on Human-Computer Interaction - INTERACT'07*, volume 4663 of *Lecture Notes in Computer Science*, pages 71–84, Rio de Janeiro, Brazil, September 2007. Springer.

[KD02] Arthur E. Kirkpatrick and Sarah A. Douglas. Application-based Evaluation of Haptic Interfaces. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS '02)*, pages 32–39, Orlando, FL, USA, March 2002. IEEE Computer Society Press.

[KFBP06] Kanav Kahol, Jamieson French, Laura Bratton, and Sethuraman Panchanathan. Learning and Perceiving Colors Haptically. In *ACM SIGACCESS Conference on Computers and Accessibility (Assets '06)*, pages 173–180, Portland, Oregon, USA, 2006. ACM Press.

[Kje00] Jesper Kjeldskov. Interaction in virtual reality. Available online at: `http://www.cs.aau.dk/~jesper/pdf/presentations/Dankomb-00-Aalborg.pdf` [Accessed: 03-01-2010], 2000. Presentation at the Danish Conference on Pattern Recognition and Image Analysis (DANCOMP'00).

[Kje01] Jesper Kjeldskov. Combining Interaction Techniques and Display Types for Virtual Reality. In R. Thomas W. Smith and M. Apperley, editors, *Conference of the Australian Computer-Human Interaction Special Interest Group (CHISIG) of the Human Factors* *and Ergonomics Society of Australia (HFESA) (OzCHI '01)*, pages 77–83, Perth, Australia, 2001. Edith Cowan University Press.

[KJR09] Liam Kurmos, Nigel W. John, and Jonathan C. Roberts. Integration of Haptics with Web3D using the SAI. In *Conference on 3D Web Technology (Web3D '09)*, pages 25–32, Darmstadt, Germany, June 2009. ACM Press.

[KR04] Michael A. Katz and Christian Rohrer. How Many Users Are Really EnoughAnd More Importantly When? Available online at: `http://www.xdstrategy.com/wp-content/uploads/2008/10/katz_and_rohrer_2004-chi.pdf` [Accessed: 03-01-2010], 2004.

[KRR09] Werner A. König, Roman Rädle, and Harald Reiterer. Squidy: A Zoomable Design Environment for Natural User Interfaces. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '09)*, pages 4561–4566, Boston, MA, USA, 2009. ACM Press.

[KSS00a] Henry König, Jochen Schneider, and Thomas Strothotte. Haptic Exploration Of Virtual Buildings Using Non-Realistic Haptic Rendering. In *International Conference on Computers Helping People with Special Needs (ICCHP'00)*, pages 377–384, Vienna, Austria, 2000. Austrian Computer Society (OCG).

[KSS00b] Henry König, Jochen Schneider, and Thomas Strothotte. Orientation and Navigation in Virtual Haptic-Only Environments. In Volker Paelke and Sabine Volbracht, editors, *Workshop on Guiding Users through Interactive Experiences: Usability Centred Design and Evaluation of Virtual 3D Environments*, pages 123–134, Paderborn, Germany, 2000. C.LAB Publication.

[KTMP05] Kanav Kahol, Priyamvada Tripathi, Troy McDaniel, and Sethuraman Panchanathan. Rendering Block Diagrams Accessible through Audio-Haptic interface. In *Workshop on Computer Vision Applications for the Visually Impaired (CVAVI'05)*, pages 8–13, San Diego, CA, USA, June 2005. IEEE Computer Society Press.

[Kur97] Martin Kurze. Rendering Drawings for Interactive Haptic Perception. In *Conference on Human Factors in Computing System (CHI*

'97), pages 423–430, Atlanta, Georgia, USA, March 1997. ACM Press.

[Kur99] Martin Kurze. Tguide: A guidance system for tactile image exploration. *Behaviour and Information Technology*, 18(1):11–17, 1999.

[LAAVM09] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt, and Benoit Macq. An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. In *ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '09)*, pages 245–254, Pittsburgh, PA, USA, July 2009. ACM Press.

[LBWR94] Gerald L. Lohse, Kevin Biolsi, Neff Walker, and Henry H. Rueter. A Classification of Visual Representations. *Communications of the ACM*, 37(12):36–49, December 1994.

[LC07] Gitte Lindgaard and Jarinee Chattratichart. Usability Testing: What Have We Overlooked? In *Conference on Human Factors in Computing Systems (CHI '07)*, pages 1415–1424, San Jose, California, USA, April 2007. ACM Press.

[LCP$^+$07] Karljohan Lundin, Matthew Cooper, Anders Persson, Daniel Evestedt, and Anders Ynnerman. Enabling design and interactive selection of haptic modes. *Virtual Reality*, 11(1):1–13, 2007.

[LD03] S. D. Laycock and A. M. Day. Recent Developments and Applications of Haptic Devices. *Computer Graphics Forum*, 22(2):117–132, 2003.

[LD04] Anatole Lécuyer and Lionel Dominjon. Techniques d'interaction avec retour haptique. Available online at: `http://lsc.univ-evry.fr/~ashaptics/docs/ASHAPTIQUE_10_06_04/Lecuyer.pdf` [Accessed: 03-01-2010], June 2004. Presentation in French.

[LD07] S.D. Laycock and A.M. Day. A Survey of Haptic Rendering Techniques. *Computer Graphics Forum*, 26(1):50–65, 2007.

[LDL$^+$04] Johnny C. Lee, Paul H. Dietz, Darren Leigh, William S. Yerazunis, and Scott E. Hudson. Haptic Pen: A Tactile Feedback Stylus for

Touch Screens. In *Symposium on User Interface Software and Technology (UIST '04)*, pages 291–294, Santa Fe, New Mexico, USA, October 2004. ACM Press.

[Lév05] Vincent Lévesque. Blindness, Technology and Haptics. Technical Report TR-CIM-05.08, Haptics Laboratory, Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada, August 2005.

[Lew06] James R. Lewis. Sample Sizes for Usability Tests:Mostly Math, Not Magic. *Interactions - Special Issue Waits & measures*, 13(6):29–33, November/December 2006.

[LG98] M. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *IMA Conference on Mathematics of Surfaces*, pages 37–56, San Diego, CA, USA, 1998.

[LH03] G. S. Lee and B. Hannaford. Anisotropies of Touch in Haptic Icon Exploration. In *Conference on Intelligent Robots and Systems (IROS '03)*, volume 3, pages 2713–2717, Las Vegas, Nevada, USA, October 2003. IEEE Computer Society Press.

[LH04] Martin Lorenz and Mirko Horstmann. Semantic Access to Graphical Web Ressources for Blind User. In *International Semantic Web Conference (ISWC '04)*, Hiroshima, Japan, November 2004. The Semantic Web Science Association (SWSA): online publication. Available online at: `http://iswc2004.semanticweb.org/posters/PID-YIHSOXCY-1090249595.pdf` [Accessed: 03-01-2010].

[LKL91] J. Bryan Lewis, Lawrence Koved, and Daniel T. Ling. Dialogue structures for virtual worlds. In *Conference on Human Factors in Computing Systems (CHI '91)*, pages 131–136, New Orleans, Louisiana, United States, 1991. ACM Press.

[LLD06] R. G. Laycock, S. D. Laycock, and A. M Day. Haptic Navigation and Exploration of High Quality Pre-rendered Environments. In *Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST'06)*, pages 17–24, Cyprus, 2006. Eurographics.

[LLPN00] Dale A. Lawrence, Christopher D. Lee, Lucy Y. Pao, and Roman Y. Novoselov. Shock and Vortex Visualization Using a Combined Visual/Haptic Interface. In *Visualization (VIS '00)*, pages 131–137, Salt Lake City, Utah, United States, 2000. IEEE Computer Society Press.

[LM00] O. Lahav and D. Mioduser. Multi-sensory virtual environment for supporting blind persons' acquisition of spatial cognitive mapping, orientation and mobility skills. In *International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT '00)*, pages 53–58, Alghero, Italy, 2000. Online proceedings, paper available at: `http://www.icdvrat.rdg.ac.uk/2004/papers/S04_N4_Lahav_ICDVRAT2004.pdf` [Accessed: 03-01-2010].

[LM03] Orly Lahav and David Mioduser. A blind person's cognitive mapping of new spaces using a haptic virtual environment. *Journal of Research in Special Education Needs*, 3(3):172–177, 2003.

[LM04] Orly Lahav and David Mioduser. Exploration of Unknown Spaces by People who are Blind using a Multi-sensory Virtual Environment. *Journal of Special Education Technology*, 19(3):15–24, 2004.

[LO08] Ming C. Lin and Miguel A. Otaduy, editors. *Haptic Rendering - Foundations, Algorithms, and Applications*. AK Peters, 2008.

[Lof03] R. B. Loftin. Multisensory Perception: Beyond the Visual in Visualization. *IEEE Computing in Science and Engineering*, 5(4):56–58, July/August 2003.

[LPLN04] Dale A. Lawrence, Lucy Y. Pao, Christopher D. Lee, and Roman Y. Novoselov. Synergistic Visual/Haptic Rendering Modes for Scientific Visualization. *IEEE Computer Graphics and Applications*, 24(6):22–30, November/December 2004.

[LPLR07] Beom-Chan Lee, Hyeshin Park, Junhun Lee, and Jeha Ryu. Tactile Visualization with MobileAR on a Handheld Device. In Ian Oakley and Stephen Brewster, editors, *Haptic and Audio Interaction Design Workshop (HAID'07)*, volume 4813 of *Lecture Notes in Computer Science*, pages 11–21, Seoul, Korea, 2007. Springer.

[LRBW90] Jerry Lohse, Henry Rueter, Kevin Biolsi, and Neff Walker. Classifying Visual Knowledge Representations: a Foundation for Visualization Research. In A. E. Kaufman, editor, *Visualization (VIS '90)*, pages 131–138, San Francisco, California, 1990. IEEE Computer Society Press.

[LRC09] Jaebong Lee, Jonghyun Ryu, and Seungmoon Choi. Vibrotactile Score: A Score Metaphor for Designing Vibrotactile Patterns. In *WorldHaptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC '09)*, pages 302–307, Salt Lake City, UT, USA, March 2009. IEEE Computer Society Press.

[LYNH06] Robert W. Lindeman, Yasuyuki Yanagida, Haruo Noma, and Kenichi Hosaka. Wearable vibrotactile systems for virtual contact and information display. *Virtual Reality*, 9(2):203–213, December 2006.

[Man97] Steve Mann. A Historical Account of 'WearComp' and 'WearCam' Inventions Developed for Applications in 'Personal Imaging'. In *International Symposium on Wearable Computers (ISWC '97)*, pages 66–73, Cambridge, Massachusetts, USA, October 1997. IEEE Computer Society Press.

[ME03] Karon MacLean and Mario Enriquez. Perceptual Design of Haptic Icons. In I. Oakley, O'Modhrain, and F. Newell, editors, *Eurohaptics (EH'03)*, pages 351–363, Dublin, Ireland, July 2003. Eurohaptics Society. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2003.shtml` [Accessed: 03-01-2010].

[MFH05] D. Mejia, P. Figueroa, and J.T. Hernandez. Interactive Support for VR Rapid Prototyping Based on InTml. In *International Workshop on Methods & Tools for Designing VR Applications (MeTo-VR)*, Ghent, Belgium, October 2005. WISE Web & Information Systems Engineering Lab.

[Min95] Mark R. Mine. Virtual Environment Interaction Techniques. Technical Report TR95-018, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, May 1995.

[MKB05] D. K. McGookin, J. Kildal, and S. A. Brewster. New Views on Haptic Graph Visualisation. In *Hands on Haptics: Exploring Non-Visual Visualisation Using the Sense of Touch (CHI '05 Workshop)*, Portland, Oregon, April 2005. Available online at: `http://www.dcs.gla.ac.uk/~johank/MultiVis-web/Publications/HandsOnHaptics(CHI2005Workshop).pdf` [Accessed 03-01-2010].

[MRG04] Charlotte Magnusson and Kirsten Rassmus-Gröhn. A Dynamic Haptic-Audio Traffic Environment. In M. Buss, M. Fritschi, and H. Esen, editors, *Eurohaptics (EH '04)*, pages 253–259, Munich, Germany, June 2004. Eurohaptics Society. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2004.shtml` [Accessed: 03-01-2010].

[MS94] T. H. Massie and J. K. Salisbury. The PHANTOM Haptic Interface: A Device for Probing Virtual Objects. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'94)*, pages 295–299, Chicago, IL, USA, November 1994. ASME. Available online at: `http://www.sensable.com/documents/documents/ASME94.pdf` [Accessed: 03-01-2010].

[MWT$^+$02] Michael C. Medlock, Dennis Wixon, Mark Terrano, Ramon L. Romero, and Bill Fulton. Using the RITE method to improve products; a definition and a case study. In *Usability Professionals Association Conference*, Florida, July 2002. UPA.

[MYN$^+$05] Erick Mendez, Shunsuke Yoshida, Haruo Noma, Robert W. Lindeman, Yasuyuki Yanagida, Shinobu Masaki, and Kenichi Hosaka. Haptic-Assisted Guidance System for Navigating Volumetric Data Sets. In *WorldHaptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05)*, pages 531–534, Pisa, Italy, 2005. IEEE Computer Society Press.

[MZ99] Timothy Miller and Robert Zeleznik. The Design of 3D Haptic Widgets. In *Symposium on Interactive 3D Graphics (SI3D '99)*, pages 97–102, Atlanta, Georgia, United States, 1999. ACM Press.

[Nes03] Keith V. Nesbitt. *Multi-sensory Display of Abstract Data.* PhD thesis, School of Information Technology, University of Sydney, NSW 2006, Australia, January 2003.

[Nes05a] K. V. Nesbitt. A Framework to Support the Designers of Haptic, Visual and Auditory Displays. In Jim Carter and David Fourney, editors, *Guidelines On Tactile and Haptic Interactions (GOTHI'05)*, pages 54–64, Saskatoon, Saskatchewan, CANADA, October 2005. USERLab, University of Saskatchewan. Available online at: `http://userlab.usask.ca/GOTHI/GOTHI-05Proceedings.html` [Accessed: 03-01-2010].

[Nes05b] Keith V. Nesbitt. Structured Guidelines to Support the Design of Haptic Displays. In Jim Carter and David Fourney, editors, *Guidelines On Tactile and Haptic Interactions (GOTHI'05)*, pages 65–74, Saskatoon, Saskatchewan, CANADA, October 2005. USERLab, University of Saskatchewan. Available online at: `http://userlab.usask.ca/GOTHI/GOTHI-05Proceedings.html` [Accessed: 03-01-2010].

[Nie00] Jakob Nielsen. Why You Only Need to Test with 5 users. Online bi-weekly column Alertbox: Current Issues in Web Usability, March 2000. Available at: `http://www.useit.com/alertbox/20000319.html` [Accessed: 03-01-2010].

[Nie03] Jakob Nielsen. Usability 101: Introduction to Usability. Online bi-weekly column Alertbox: Current Issues in Web Usability, August 2003. Available at: `http://www.useit.com/alertbox/20030825.html` [Accessed: 03-01-2010].

[Nie06] Jakob Nielsen. Quantitative Studies: How Many Users to Test? Online bi-weekly column Alertbox: Current Issues in Web Usability, June 2006. Available at: `http://www.useit.com/alertbox/quantitative_testing.html` [Accessed: 03-01-2010].

[NL93] Jakob Nielsen and Thomas K. Landauer. A Mathematical Model of the Finding of Usability Problems. In *INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (INTERCHI*

'93), pages 206–213, Amsterdam, The Netherlands, 1993. ACM Press.

[NMF$^+$05] Jessie Y. C. Ng, Jo C. F. Man, Sidney Fels, Guy Dumont, and J. Mark Ansermino. An Evaluation of a Vibro-Tactile Display Prototype for Physiological Monitoring. *Anesthesia & Analgesia*, 101(6):1719–1724, December 2005.

[NPDB06] David Navarre, Philippe Palanque, Pierre Dragicevic, and Rémi Bastide. An approach integrating two complementary model-based environments for the construction of multimodal interactive applications. *Interacting with Computers*, 18(5):910–941, 2006.

[Osa06] Noritaka Osawa. Tactile Glyphs for Palpation of Relationships. In *Information Visualization (IV '06)*, pages 575–584, London, England, July 2006. IEEE Computer Society Press.

[Ovi99] Sharon Oviatt. Ten Myths of Multimodal Interaction. *Communications of ACM*, 42(11):74–81, November 1999.

[Owe99] G. Scott Owen. Definitions and Rationale for Visualization. Online at: `http://www.siggraph.org/education/materials/HyperVis/visgoals/visgoal2.htm` [Accessed: 03-01-2010], February 1999. HyperVis project from SIGGRAPH.

[PB03] P. Parente and G. Bishop. BATS: The Blind Audio Tactile Mapping System. In *ACM South Eastern Conference (ACMSE '03)*, Savannah, GA., March 2003. ACM Press. Available online at: `http://www.cs.unc.edu/Research/assist/bats/papers/BATS.pdf` [Accessed: 03-01-2010].

[Pet67] D. L. Peterson. Computer-Controlled Tactile Display. Master's thesis, Department of Electronic Engineering, MIT, University of Purdue, September 1967.

[PF09] Karljohan Lundin Palmerius and Camilla Forsell. The Impact of Feedback Design in Haptic Volume Visualization. In *WorldHaptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC*

'09), pages 154–159, Salt Lake City, UT, USA, March 2009. IEEE Computer Society Press.

[PL98] L. Y. Pao and D. A. Lawrence. Synergistic Visual/Haptic Computer Interfaces. In *Japan/USA/Vietnam Workshop on Research and Education in Systems, Computation, and Control Engineering (RESCCE '98)*, pages 155–162, Hanoi, Vietnam, 1998.

[PLLM06] Jérome Pasquero, Joseph Luk, Shannon Little, and Karon MacLean. Perceptual Analysis of Haptic Icons: an Investigation into the Validity of Cluster Sorted MDS. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'06)*, pages 67–74, Alexandria, Virginia, March 2006. IEEE Computer Society Press.

[PR07] Sabrina Panëels and Jonathan C. Roberts. Haptic Guided Visualization of Line Graphs: Pilot Study. In Lorna Brown and Tae-Jeong Jang, editors, *Workshop on Haptic and Audio Interaction Design (HAID '07), poster and demo proceedings*, pages 5–6, Seoul, Korea, November 2007.

[PR09] Sabrina Panëels and Jonathan C. Roberts. Review of Designs for Haptic Data Visualization. *Transactions on Haptics*, 2009. PrePrint (to appear in the April-June 2010 issue).

[PRR09] Sabrina Panëels, Jonathan C. Roberts, and Peter J. Rodgers. Haptic Interaction Techniques for Exploring Chart Data. In M. Ercan Altinsoy, Ute Jekosch, and Stephen Brewster, editors, *Workshop on Haptic and Audio Interaction Design (HAID '09)*, volume 5763 of *Lecture Notes in Computer Science*, pages 31–40, Dresden, Germany, September 2009. Springer.

[PRR10] Sabrina Panëels, Jonathan C. Roberts, and Peter J. Rodgers. HITPROTO: a Tool for the Rapid Prototyping of Haptic Interactions for Haptic Data Visualization. In *Haptics Symposium (previously known as the Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems) (HAPTICS '10)*. IEEE Computer Society Press, 2010.

[PS03] Ludek Pokluda and Jiri Sochor. Spatial Haptic Orientation for Visually Impaired People. In I. Oakley, S. O'Modhrain, and F. Newell, editors, *Eurographics Ireland Chapter Workshop Series*, volume 2, pages 29–34, University of Ulster, Coleraine, United Kingdom, April 2003. Eurographics.

[PS05] Ludek Pokluda and Jiri Sochor. Spatial Orientation in Buildings Using Models with Haptic Feedback. In *WorldHaptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05)*, pages 523–524, Pisa, Italy, 2005. IEEE Computer Society Press.

[RB07] Andrew Ray and Doug A. Bowman. Towards a System for Reusable 3D Interaction Techniques. In *Symposium on Virtual Reality Software and Technology (VRST '07)*, pages 187–190, Newport Beach, California, 2007. ACM Press.

[RF05] J. C. Roberts and K. Franklin. Haptic Glyphs (Hlyphs) - structured haptic objects for haptic visualization. In Antonio Bicchi and Massimo Bergamasco, editors, *WorldHaptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05)*, pages 369–374, Pisa, Italy, March 2005. IEEE Computer Society Press.

[RFC02] Jonathan C. Roberts, Keith Franklin, and Jonathan Cullinane. Virtual Haptic Exploratory Visualization of Line Graphs and Charts. In Andrew J. Woods, John O. Merritt, Stephen A. Benton, and Mark T. Bolas, editors, *The Engineering Reality of Virtual Reality*, volume 4660B of *Electronic Imaging Conference*, pages 401–410. IS&T/SPIE, January 2002.

[Rhi10] Rhino, a NURBS-based 3D modeling tool. `http://www.rhino3d.com/` [Accessed: 03-01-2010], 2010.

[Rob00] Jonathan C. Roberts. Visualization Display Models - ways to classify visual representations. *International Journal of Computer Integrated Design and Construction*, 2(4):241–250, December 2000.

[Rob04] Jonathan C. Roberts. Visualization Equivalence for Multisensory Perception. *IEEE Computing in Science & Engineering*, 6(3):61–65, May/June 2004.

[RP07] Jonathan C. Roberts and Sabrina Panëels. Where are we with Haptic Visualization? In *WorldHaptics - Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC '07)*, pages 316–323, Tsukuba, Japan, March 2007. IEEE Computer Society Press.

[RTW05] M. Rossi, K. Tuer, and D. Wang. A New Design Paradigm for the Rapid Development of Haptic and Telehaptic Applications. In *Conference on Control Applications (CCA '05)*, pages 1246–1250, Toronto, Canada, August 2005. IEEE Computer Society Press.

[Rub94] Jeffrey Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests.* Wiley Technical Communications Library. John Wiley and Sons, Inc., April 1994.

[Rup00] A. H. Rupert. An for Instrumentation Solution Reducing Spatial Disorientation Mishaps. *IEEE Engineering in Medicine and Biology*, 19(2):71–80, March/April 2000.

[RYB$^+$00] R. Ramloll, W. Yu, S. Brewster, B. Riedel, M. Burton, and G. Dimigen. Constructing Sonified Haptic Line Graphs for the Blind Student: First Steps. In *ACM SIGACCESS Conference on Computers and Accessibility (Assets '00)*, pages 17–25, Arlington, Virginia, United States, 2000. ACM Press.

[SB97] M. A. Srinivasan and C. Basdogan. Haptics in Virtual Environments: Taxonomy, Research status, and Challenges. *Computers and Graphics*, 21(4):393–404, July 1997.

[SCB04] Kenneth Salisbury, François Conti, and Federico Barbagli. Haptic Rendering: Introductory Concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, March/April 2004.

[Sen10] The SensAble Technologies. `http://www.sensable.com/company-about-us.htm` [Accessed: 03-01-2010], 2010.

[Shn96] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Symposium on Visual Languages (VL '96)*, page 336, Boulder, Colorado, USA, September 1996. IEEE Computer Society Press.

[SJGL08] Orit Shaer, Robert J. K. Jacob, Mark Green, and Kris Luyten, editors. *CHI Workshop on User Interface Description Languages for Next Generation User Interfaces*, Florence, Italy, April 2008. ACM Press. Available online at: `http://www.eecs.tufts.edu/~oshaer/workshop/` [Accessed: 03-01-2010].

[SJGL10] Orit Shaer, Robert J.K. Jacob, Mark Green, and Kris Luyten, editors. *ACM Transactions on Computer-Human Interaction Special Issue: User Interface Description Languages for Next Generation User Interfaces*. ACM Press, 2010. To appear in Vol. 16, Issue 4, call available at: `http://tochi.acm.org/si/uidl.shtml` [Accessed: 03-01-2010].

[SJN08] Marcos Serrano, David Juras, and Laurence Nigay. A Three-dimensional Characterization Space of Software Components for Rapidly Developing Multimodal Interfaces. In *Conference on Multimodal interfaces (IMCI '08)*, pages 149–156, Chania, Crete, Greece, October 2008. ACM Press.

[Sjö01a] Calle Sjöström. Designing Haptic Computer Interfaces for Blind People. In *International Symposium on Signal Processing and its Applications (ISSPA 2001)*, pages 68–71, Kuala Lumpur, Malaysia, August 2001. IEEE Computer Society Press.

[Sjö01b] Calle Sjöström. Virtual Haptic Search Tools - The White Cane in a Haptic Computer Interface. In *Association for the Advancement of Assistive Technology in Europe Conference (AAATE '01)*, pages 124–128, Ljubljana, Slovenia, September 2001. IOS Press, Amsterdam.

[Sjö02] Calle Sjöström. *Non-Visual Haptic Interaction Design - Guidelines and Applications*. PhD thesis, Certec, Lund University, Sweden, 2002.

[SM05] Harri Siirtola and Erkki Mäkinen. Constructing and reconstructing the reorderable matrix. *Information Visualization*, 4(1):32–48, 2005.

[SMMC06] C. Swindells, E. Maksakov, K. E. MacLean, and V. Chung. The Role of Prototyping Tools for Haptic Behavior Design. In *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS '06)*, pages 161–168, Alexandria, Virginia, USA, March 2006. IEEE Computer Society Press.

[SNL$^+$08] Marcos Serrano, Laurence Nigay, Jean-Yves L. Lawson, Andrew Ramsay, Roderick Murray-Smith, and Sebastian Denef. The OpenInterface Framework: a Tool for Multimodal Interaction. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '08)*, pages 3501–3506, Florence, Italy, April 2008. ACM Press.

[SP09] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 5th edition, 2009.

[Sri04] Mandayam A. Srinivasan. What is Haptics? Available online at: `http://www.sensable.com/documents/documents/what_is_haptics.pdf` [Accessed: 03-01-2010], 2004.

[SS00] Jochen Schneider and Thomas Strothotte. Constructive Exploration of Spatial Information by Blind Users. In *ACM SIGACCESS Conference on Computers and Accessibility (Assets '00)*, pages 188–192, Arlington, Virginia, USA, 2000. ACM Press.

[SS01] Jared Spool and Will Schroeder. Testing Web Sites: Five Users Is Nowhere Near Enough. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '01)*, pages 285–286, Seattle, Washington, 2001. ACM Press.

[Sto00] Robert J. Stone. Haptic Feedback: A Potted History, From Telepresence to Virtual Reality. In S. Brewster and R. Murray-Smith, editors, *Workshop on Haptic Human-Computer Interaction*, volume 2058 of *Lecture Notes in Computer Science*, pages 1–7, Glasgow, UK, 2000. Springer.

[SW03] Christian Springsguth and Gerhard Weber. Design Issues of Relief Maps for Haptic Displays. In C. Stephanidis, editor, *International Conference on Human-Computer Interaction (HCI '03)*, volume 4, pages 1477–1481, Crete, June 2003. Lawrence Erlbaum. Available online at: `http://www.multireader.org/design.htm` [Accessed: 03-01-2010].

[SWS+03] Ganesh Sankaranarayanan, Suzanne Weghorst, Michel Sanner, Alexandre Gillet, and Arthur Olson. Role of Haptics in Teaching Structural Molecular Biology. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, pages 363–367, Los Angeles, CA, USA, March 2003. IEEE Computer Society Press.

[TBL02] C. L. Teo, E. Burdet, and H. P. Lim. A Robotic Teacher of Chinese Handwriting. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS '02)*, pages 335–341, Orlando, FL, USA, March 2002. IEEE Computer Society Press.

[TGYT03] Hong Z. Tan, Robert Gray, J. Jay Young, and Ryan Traylor. A Haptic Back Display for Attentional and Directional Cueing. *Haptics-e*, 3(1):1–20, June 2003. Available online at: `http://www.haptics-e.org/Vol_03/he-v3n1.pdf` [Accessed: 03-01-2010].

[TI00] Russell M. Taylor II. Practical Sciontific Visualization Examples. *Comput. Graphics*, 34(1):74–79, 2000.

[TICO+97] Russell M. Taylor II, Jun Chen, Shoji Okimoto, Noel Llopis-Artime, Vernon L. Chi, Frederick P. Brooks, Jr., Mike Falvo, Scott Paulson, Pichet Thiansathaporn, David Glick, Sean Washburn, and Richard Superfine. Pearls Found on the way to the Ideal Interface for Scanned-probe Microscopes. In *Visualization (VIS '97)*, pages 467–470, Phoenix, Arizona, USA, October 1997. IEEE Computer Society Press.

[TIRC+93] Russell M. Taylor II, Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Erik J. Snyder. The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope. In *Conference on Computer*

Graphics and Interactive Techniques (SIGGRAPH '93), pages 127–134, Anaheim, CA, USA, 1993. ACM Press.

[TM04] Melanie Tory and Torsten Möller. Rethinking Visualization: A High-Level Taxonomy. In *IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 151–158. IEEE Computer Society Press, 2004.

[TP01] Hong Z. Tan and Alex Pentland. Tactual Displays for Sensory Substitution and Wearable Computers. In Woodrow Barfield and Thomas Caudell, editors, *Fundamentals of Wearable Computers and Augmented Reality*, pages 579–598. Lawrence Erlbaum Associates, Mahwah, NJ, 2001.

[TRR08] Sampo Töyssy, Jukka Raisamo, and Roope Raisamo. Telling Time by Vibration. In Manuel Ferre, editor, *Eurohaptics (EH '08)*, volume 5024 of *Lecture Notes in Computer Science*, pages 924–929. Springer, 2008.

[Usa10] Usability.gov. Usability Basics. `http://www.usability.gov/basics/index.html` [Accessed: 03-01-2010], 2010.

[VBM09] Ramiro Velázquez, Omar Bazán, and Marco Magana. A Shoe-Integrated Tactile Display for Directional Navigation. In *Conference on Intelligent RObots and Systems (IROS'09)*, pages 1235–1240, St. Louis, USA, October 2009. IEEE Computer Society Press.

[vE02] Jan B.F. van Erp. Guidelines for the Use of Vibro-Tactile Displays in Human Computer Interaction. In S. A. Wall, B. Riedel, A. Crossan, and M. R. McGee, editors, *Eurohaptics (EH'02)*, pages 18–22, Edinburgh, UK, 2002. Eurohaptics Society. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2002.shtml` [Accessed: 03-01-2010].

[vEAC06] Jan B.F. van Erp, Ian Andrew, and Jim Carter. ISO's Work on Tactile and Haptic Interaction Guidelines. In *Eurohaptics (EH'06)*, pages 467–470, Paris, France, July 2006. Eurohaptics Society.

[vEBC04] M. P. van Esch-Bussemakers and A. H. M. Cremers. User Walkthrough of Multimodal Access to Multidimensional Databases.

In *International Conference on Multimodal Interfaces (ICMI'04)*, pages 220–226, State College, PA, USA, 2004. ACM Press.

[vECN$^+$06] Jan van Erp, Jim Carter, Keith Nesbitt, Ian Andrew, David Fourney, Shinichi Fukuzumi, Gunnar Jansson, Helmut Jrgensen, Roland Kadefors, Tadashi Kobayashi, Misa Grace Kwok, and Stephen Brewster. Ergonomics of human-system interaction - Guidance on tactile and haptic interactions (Draft for ISO International Standard). Available online at: `http://ergonomics.org.uk/espdfs/PH_9-0171_06%20Haptic%20interactions.pdf` [Accessed: 02-01-2010], November 2006.

[vEK08] Jan B.F. van Erp and Thorsten A. Kern. ISOs Work on Guidance for Haptic and Tactile Interactions. In M. Ferre, editor, *Eurohaptics (EH'08)*, volume 5024 of *Lecture Notes in Computer Science*, pages 936–940. Springer, 2008.

[vEvV01] Jan B. F. van Erp and Hendrik A. H. C. van Veen. Vibro-Tactile Information Presentation in Automobiles. In C. Baber, M. Faint, S. Wall, and Wing A.M., editors, *Eurohaptics (EH'01)*, pages 99–104, Birmingham, UK, 2001. Eurohaptics Society. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2001.shtml` [Accessed: 03-01-2010].

[vEvVJD05] Jan B. F. van Erp, Hendrik A. H. C. van Veen, Chris Jansen, and Trevor Dobbins. Waypoint Navigation with a Vibrotactile Waist Belt. *ACM Transactions on Applied Perception*, 2(2):106–117, April 2005.

[VLC09] Yon Visell, Alvin Law, and Jeremy R. Cooperstock. Toward Iconic Vibrotactile Information Display Using Floor Surfaces. In *World-Haptics - Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC '09)*, pages 267–272, Salt Lake City, USA, March 2009. IEEE Computer Society Press.

[vRBKB03] T. van Reimersdahl, F. Bley, T. Kuhlen, and C. H. Bischof. Haptic Rendering Techniques for the Interactive Exploration of CFD Datasets in Virtual Environments. In J. Deisinger and A. Kunz, editors, *Eurographics Workshop on Virtual Environments (EGVE '03)*, pages 241–246, Zurich, Switzerland, 2003. ACM Press.

[vVvE00] Henricus A. H. C van Veen and Jan B. F. van Erp. Tactile Information Presentation in the Cockpit. In S. Brewster and R. Murray-Smith, editors, *Workshop on Haptic Human-Computer Interaction*, volume 2058 of *Lecture Notes in Computer Science*, pages 174–181, Glasgow, UK, 2000. Springer.

[Wal05] S.A. Wall. Using More Salient Haptic Cues in Data Visualisation. In *Hands on Haptics: Exploring Non-Visual Visualisation Using the Sense of Touch (CHI '05 Workshop)*, Portland, Oregon, April 2005. Available online at: `http://www.dcs.gla.ac.uk/haptic/haptics%20web%20pages_files/Wall.pdf` [Accessed: 03-01-2010].

[WB97a] T.P Way and K. Barner. Automatic Visual to Tactile Translation, Part I: Human Factors, Access Methods and Image Manipulation. *IEEE Transactions on Rehabilitation Engineering*, 5(1):81–94, 1997.

[WB97b] T.P Way and K. Barner. Automatic Visual to Tactile Translation, Part II: Evaluation of the TACTile Image Creation System. *IEEE Transactions on Rehabilitation Engineering*, 5(1):95–105, 1997.

[WB03] S. A. Wall and S. A. Brewster. Assessing Haptic Properties for Data Representation. In *Conference on Human Factors in Computing Systems - Extended Abstracts (CHI EA '03)*, volume 2, pages 858–859, Ft. Lauderdale, Florida, USA, 2003. ACM Press.

[WB04] S.A. Wall and S.A. Brewster. Providing External Memory Aids in Haptic Visualisations for Blind Computer Users. In *International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT)*, pages 157–164, New College, Oxford, 2004. Also published in the International Journal of Disability and Human Development, volume 4(4), 2005, pages 331–338. Available online at: `http://www.dcs.gla.ac.uk/~johank/MultiVis-web/Publications/IJDHD_wall.pdf` [Accessed: 03-01-2010].

[WB06a] Steven Wall and Stephen Brewster. Feeling What You Hear: Tactile Feedback for Navigation of Audio Graphs. In *Conference on Human Factors in Computing Systems (CHI '06)*, pages 1123–1132, Montréal, Québec, Canada, April 2006. ACM Press.

[WB06b] Steven A. Wall and Stephen A. Brewster. Tac-tiles: Multimodal Pie Charts for Visually Impaired Users. In *Nordic Conference on Human-Computer Interaction (NordiCHI '06)*, pages 9–18, Oslo, Norway, October 2006. ACM Press.

[WC01] Alan Woolrych and Gilbert Cockton. Why and When Five Test Users aren't Enough. In J. Vanderdonckt, A. Blandford, and A. Derycke, editors, *Journes Francophones sur l'Interaction Homme-Machine (IHM) (French-speaking Meeting on Human-Machine Interaction) (IHM-HCI '01)*, volume 2, pages 105–108. Cepadues-Editions, Toulouse, France 2001.

[Web10] Web3D Consortium. The X3D Standard. `http://www.web3d.org/about/overview/` [Accessed: 03-01-2010], 2010.

[WGO$^+$00] Evan F. Wies, John A. Gardner, M. Sile O'Modhrain, Christopher J. Hasser, and Vladimir L. Bulatov. Web-Based Touch Display for Accessible Science Education. In *Workshop on Haptic Human-Computer Interaction*, volume 2058 of *Lecture Notes in Computer Science*, pages 52–60. Springer, 2000.

[WL90] Stephen Wehrend and Clayton Lewis. A Problem-oriented Classification of Visualization Techniques. In *Visualization '90 (VIS '90)*, pages 139–143, San Francisco, California, 1990. IEEE Computer Society Press.

[Xmd10] XmdvTool Project Homepage. Iris DataSet. `http://davis.wpi.edu/~xmdv/datasets/iris.html` [Accessed: 03-01-2010], 2010.

[YB02a] W. Yu and S. Brewster. Comparing Two Haptic Interfaces for Multimodal Graph Rendering. In *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems (HAPTICS '02)*, pages 3–9, Florida, USA, 2002. IEEE Computer Society Press.

[YB02b] W. Yu and S. Brewster. Multimodal Virtual Reality Versus Printed Medium in Visualization for Blind People. In *ACM SIGACCESS Conference on Computers and Accessibility (Assets '02)*, pages 57–64, Edinburgh, Scotland, 2002. ACM Press.

[YB03] W. Yu and S. A. Brewster. Evaluation of Multimodal Graphs for Blind People. *Universal Access in the Information Society*, 2(2):105–124, June 2003.

[YCB02] W. Yu, K. Cheung, and S.A. Brewster. Automatic Online Haptic Graph Construction. In S.A. Wall, B. Riedel, A. Crossan, and M. R. McGee, editors, *Eurohaptics (EH '02)*, pages 128–133, Edinburgh, UK, 2002. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2002/yu.pdf` [Accessed: 03-01-2010].

[YGB01] W. Yu, K. Guffie, and S. Brewster. Image to Haptic Data Conversion: A First Step to Improving Blind Peoples Accessibility to Printed Graphs. In C. Baber, M. Faint, S. Wall, and A. M. Wing, editors, *Eurohaptics (EH '01)*, Birmingham, UK, 2001. Available online at: `http://www.eurohaptics.vision.ee.ethz.ch/2001/yu.pdf` [Accessed: 03-01-2010].

[YH02] Dingrong Yi and Vincent Hayward. Augmenting Computer Graphics with Haptics for the Visualization of Vessel Networks. In *Pacific Conference on Computer Graphics and Applications (PG '02)*, pages 375–384. IEEE Computer Society Press, 2002.

[YJN$^+$96] Christine Youngblut, Rob E. Johnson, Sarah H. Nash, Ruth A. Wienclaw, and Craig A. Will. Review of virtual environment interface technology. Technical Report IDA Paper P-3186, Institue for Defense Analyses (IDA), March 1996.

[YRB00] W. Yu, R. Ramloll, and S. Brewster. Haptic Graphs for Blind Computer Users. In S. Brewster and R. Murray-Smith, editors, *Workshop on Haptic Human-Computer Interaction*, volume 2058 of *Lecture Notes in Computer Science*, pages 102–107, Glasgow, UK, 2000. Springer.

[YRBR01] W. Yu, R. Ramloll, S. A. Brewster, and B. Riedel. Exploring computer-generated line graphs through virtual touch. In *International Symposium on Signal Processing and its Applications (ISSPA '01)*, pages 72–75, Kuala-Lampur, Malaysia, August 2001. IEEE Computer Society Press.

[ZLB+87] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A Hand Gesture Interface Device. In *SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface (CHI + GI '87)*, pages 189–192, Toronto, Ontario, Canada, 1987. ACM Press.

Appendix A

Line Chart Application: Pilot Study Materials

A.1 Pilot Study Script

Experiment: The application will show you an engraved line graph attached to a solid background as well as raised axes. The experiment will evaluate three conditions: the free exploration of the graph (key press 5), which means you are moving freely around the graph to feel the background, axes and the line; the fully constrained guided exploration (key press 4) where you are guided along the line and that highlights the points of interests (min, max, intersections) by slowing down at those points (it's the most constrained tour) and guidance (in the form of other guided tours) coupled with free exploration. The musem tour (key press 1) stops at points of interest for a short time and enables you to move around in a certain range to explore the close vicinity of the point of interest. The water skier tour (key press 2) just slows down at points of interest and enables you to move sides to sides. The last tour mode just helps you stay on the line but enables you to move freely on the line.
So to summarize:

- Free exploration - key 5
- Fully guided exploration - key 4
- Other guided tours: museum tour (key 1), water skier tour (key 2), help to stay on the line (key 3)

- The tours (except from the help to stay on the line) guide you to the origin then along the x-axis and then along the y-axis to the first point of the line, where it stops for a bit before starting the tour. (be aware that the first point can also be a point of interest and then the stopping time can be longer/or slowing down).

Tasks:

1. You will first have some practise time with some line charts to familiarise with the haptic device, the application and the different modes.

2. Then you will be asked to explore the line with different levels of complexity (easy, medium, hard) with a fixed time in either mode. For the guided tours mode, trying all tours in order 1-2-3 is highly recommended with free exploration but the choice is left to the user. However the user should at least use the museum tour (key 1) and free exploration. The main exploration objective is to understand the line layout (relative to the axes) and its points of interest (location and occurences).

3. Afterwards, you will have to answer the following questions which consist in giving the number of points of interest counted and drawing the general shape of the line graph and locate on it the points of interest (min, max, intersections) within a time limit.

4. You will also be asked to rate some areas concerning the effectiveness of the application as well as giving some comments.

NB: during the guidance, please hold the haptic device not perpendicular to the line and in a flexible way so that the resistance to the movement exerted is not too powerful.

A.2 Pilot Study Tasks

Graph:

Condition tested:

Time:

Questions:

1. Number of minimum values:
2. Number of maximum values:
3. Number of intersections with X axis:
4. Number of intersections with Y axis:
5. Draw the **general shape** of the line graph and **locate** on it **minimum and maximum values** as well as **intersections with the axes** if any. (Draw the square with each quadrant).

A.3 Pilot Study Questionnaire

Name:

Date:

Experience with FF devices or haptics in general:

Experience with Computer Science:

1. Rate the following areas:
   - Overall effectiveness of the guided tours (1 not effective at all, useless - 5 very effective very useful)
   - Effectiveness of each of the tours you have tried, namely Museum tour, Water Skier and Magnetic line (same scale as before)
   - Perceived difficulty of the task (1 very hard - 5 very easy)
   - Workload of the keyboard use/tours associated to keyboard (1 very high workload - 5 very low workload)
   - Usefulness of axes, engraved line (1 totally useless - 5 very useful)
2. Cite drawbacks
3. Cite good points
4. Was it easy to navigate?
5. Was it easy to locate specific points of interest?
6. Are there some elements you would have liked to be available in the interface?
7. Were the instructions easy to understand?
8. Additional comments:

Appendix B

Usability Evaluation Materials

B.1 Experiment Script

Welcome and Purpose

Thank you for agreeing to participate in this evaluation. Today we are asking you to complete a set of scenarios. We will record your actions on the computer, reactions and opinions using an audio recorder.

Test Facilitator's Role

I am here to record your reactions to the tool you will use. Please attempt the problems without my help. You can however ask me any questions you would like during the training session.

Role

In this experiment, you will represent a designer or a teacher, who wants to design and test some force-feedback interactions.

Experiment Setup

1. *Information:* you will be asked to fill out a form with some information about you (experience with the technology and with programming tools).

2. *Training:* you will first be introduced to what haptics is through brief demonstrations to familiarize yourself with the technology and the device setup. Using the prototyping tool will be explained by a step-by-step tutorial. I will be reading the tutorial to you and you will be using the tool accordingly. At the end of the tutorial, you will be given a "check yourself"

example to create a diagram on your own, which solution will be then given and explained. In this phase, you can ask me any questions about the technology or the tool. After you completed the training, you will be able to take a short break before starting the evaluation.

3. *Tasks:* you will be given a set of "interaction scenarios" that will describe the interactions you are trying to design. You will have to create the corresponding diagrams of these interactions using the prototyping tool. You will be able to test these diagrams until you decide they fulfil their given purpose, in which case you will notify me, save the diagram and move onto the next scenario. During this phase, you will not be allowed to communicate with me, except for serious trouble with the technology or the tool or if you do not understand the description of the interaction scenario. The time you take to produce the correct interaction diagram will be measured. However, please bear in mind that getting a diagram that produces the intended interaction is the most important. Therefore producing a correct diagram should be your goal rather than producing it quickly.

4. *Questionnaire:* : once you completed all the tasks, you will be asked to complete a questionnaire about your experience with the technology and the tool. Any comments are welcome, including criticism. At the end, there will be a brief interview based on the questionnaire, to get more details about your experience with the system.

Payment and Duration: you will be paid 10 pounds for your participation. This experiment will last up to 2 hours.

Things to Keep in Mind

Here are some things that you should know about your participation:

- This is not a test of you; you are testing the tool. So do not worry about making mistakes.

- If you ever feel that you are lost or cannot complete a scenario with the information that you have been given, please let me know. I will either put you on the right track or move you on to the next scenario.

Do you have any questions before we begin?

B.2 Background Questionnaire

Name:

Age:

Subject studied:

Left or right-handed:

PRODUCT EXPERIENCE:

1. Have you ever heard of haptics or force-feedback technologies before? Haptic technology includes devices providing tactile feedback (i.e. vibrations) or force feedback and gives the feeling of touch.
   ____ Yes ____ No

2. Have you ever used haptics or force-feedback technologies before?
   ____ Yes ____ No

   If yes, please give a list of the devices you have used.

3. What is your experience with programming? Please rate it circling the right value and give a list of languages used if applicable:
   0 - I've never programmed in my life
   1 - I have done a little bit of programming
   2 - I have programmed quite a lot
   3 - I'm a programmer and know many languages

4. Have you ever used visual programming tools before? Visual programming tools are tools that allow you to create programs by manipulating program elements graphically rather than by specifying them textually. The program elements are often represented through boxes and linked through arrows.
   ____ Yes ____ No

   If yes, please give a list of the tools you have used.

5. How long have you been using personal computers?

6. On a typical work day, how often do you use a computer to perform your job?

B.3 Consent Form

Title of project: The HITPROTO tool

Name of Researcher: Sabrina Panëels

1. I confirm that I have read and understood the information for this study and I have had the opportunity to ask questions. ☐

2. I confirm that I have understood that my actions on the computer and the conversations with the researcher will be recorded during the study. ☐

3. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason, without my counselling care or legal rights being affected. ☐

4. I agree to take part in this study ☐

Name of the Participant: ________________________________

Signature: ______________________ Date: ________________

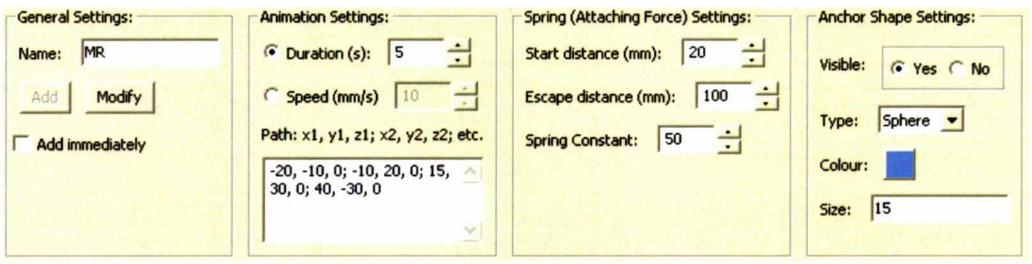
Please retain this copy for your records

B.4 Blocks Summary

ACTION BLOCKS

Stop: compulsory block that delimitates the end of the 'interaction scenario' or program. There are no parameters. Several lines can be connected to it, but none can depart from it.

Guidance_Add: creates (and can add to the scene) a guidance instance, that needs to be named. A guidance instance includes a spring to attach the device, an anchor to visualize the spring and parameters such as path and speed/duration. See the user guide pages 14-16 for a more detailed description of the parameters.



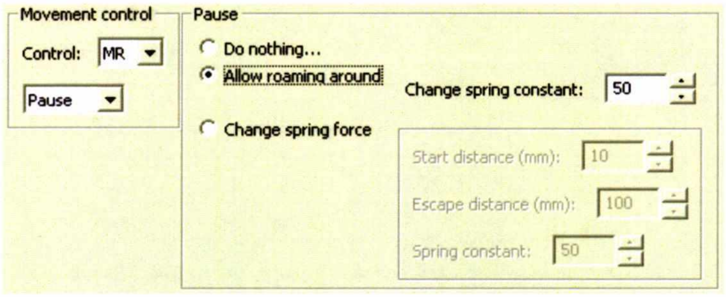
General Settings:
Name: MR
Add | Modify
☐ Add immediately

Animation Settings:
[✓ Duration (s)]: 5
Speed (mm/s): 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 15

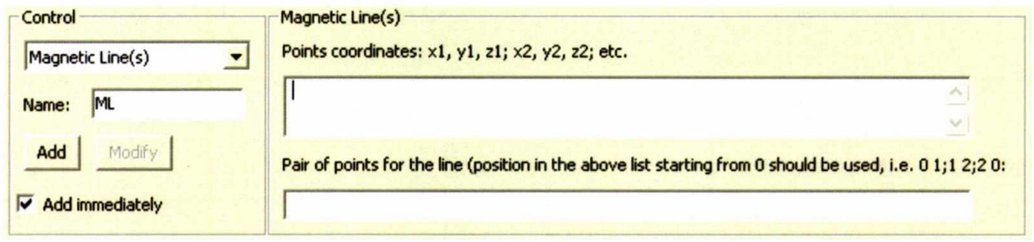
Guidance_Control: enables to control the guidance instance using its name, by starting, pausing, resuming (after a set time) or stopping it. See the user guide pages 16-17 for a more detailed description of the parameters.



Movement control
Control: MR
Pause

Pause
Do nothing...
[✓ Allow roaming around]
Change spring force
Change spring constant: 50
Start distance (mm): 10
Escape distance (mm): 100
Spring constant: 50

Haptic Effect: creates (and can add to the scene) a chosen haptic effect, which needs to be named. The available haptic effects are: SpringEffect, Magnetic Line(s) and PositionFunctionEffect. See the user guide pages 17-18 for a more detailed description of the parameters.


Control

Magnetic Line(s)

Name: ML

Add Modify

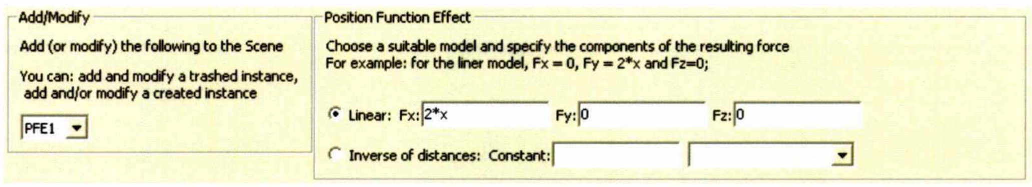
☑ Add immediately

Magnetic Line(s)

Points coordinates: x1, y1, z1; x2, y2, z2; etc.

Pair of points for the line (position in the above list starting from 0 should be used, i.e. 0 1;1 2;2 0:

Add_Modify: allows the addition of a previously removed object, the addition of a created but not yet added instance or the modification of an existing instance using its name. See the user guide page 20 for a more detailed description of the parameters.


Add/Modify

Add (or modify) the following to the Scene

You can: add and modify a trashed instance, add and/or modify a created instance

PFE1

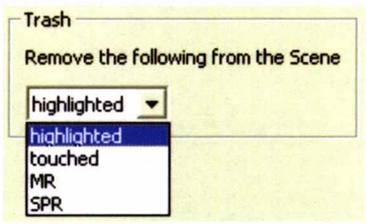
Position Function Effect

Choose a suitable model and specify the components of the resulting force
For example: for the liner model, Fx = 0, Fy = 2*x and Fz=0;

● Linear: Fx: 2*x Fy: 0 Fz: 0

○ Inverse of distances: Constant:

Trash: enables the removal of an object using its name. The removal does not delete the object itself, which can be re-added using the Add_Modify block. See the user guide page 18 for a more detailed description of the parameters.

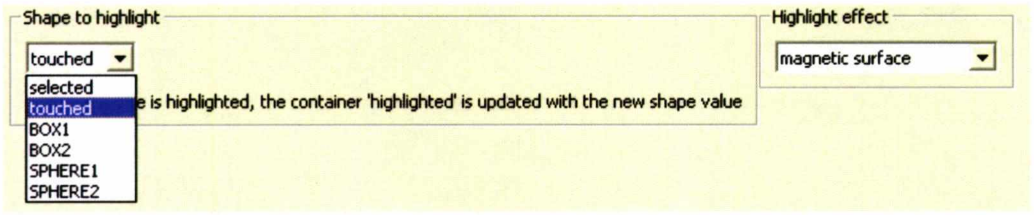

Trash

Remove the following from the Scene

highlighted

highlighted
touched
MR
SPR

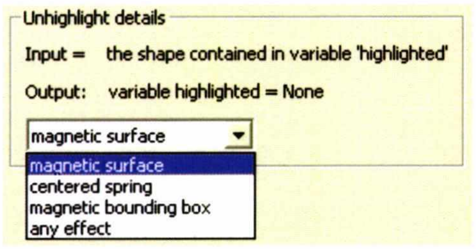
Highlight: enables to "highlight" haptically a named object from the scene by adding a spring to it, making it magnetic or surrounding it with a magnetic bounding box. See the user guide page 20 for a more detailed description of the parameters.


Shape to highlight: touched

…e is highlighted, the container 'highlighted' is updated with the new shape value

Highlight effect: magnetic surface

Unhighlight: enables to remove the highlighting effect. The option 'any effect' can be chosen in cases where multiple effects have been used. See the user guide page 21 for a more detailed description of the parameters.

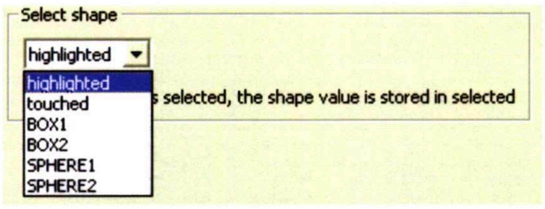

Unhighlight details

Input = the shape contained in variable 'highlighted'

Output: variable highlighted = None

magnetic surface

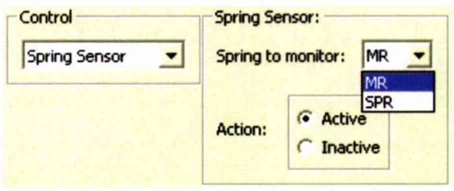
Select: enables to keep track of the selected object by putting the name of the object into memory by storing it in the selected name. See the user guide page 22 for a more detailed description of the parameters.

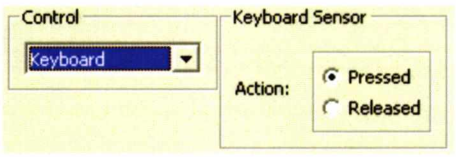

Select shape: highlighted

…s selected, the shape value is stored in selected

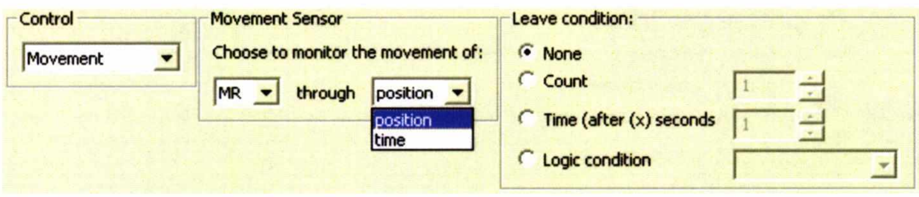
FLOW BLOCKS

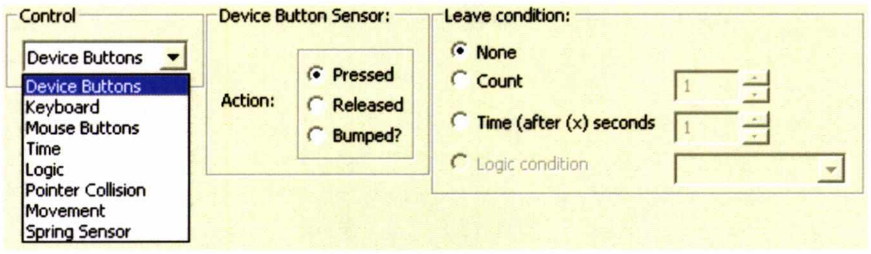
Wait For: enables the interruption of the sequence of actions until a chosen event happens. The events can be: the device button being pressed/released, a keyboard key being pressed/released, a mouse button being pressed/released, an elapsed time or the activation of a spring (from a guidance or a haptic effect instance). See the user guide page 8 for a more detailed description of the parameters.

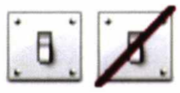

Control: Spring Sensor
Spring Sensor:
Spring to monitor: MR (MR, SPR)
Action: [✓ Active] Inactive


Control: Keyboard
Keyboard Sensor
Action: [✓ Pressed] Released

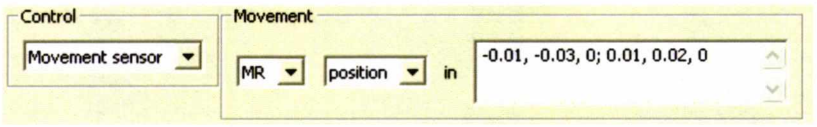
Everytime and **Everytime_end:** enables to execute a set of actions specified within the two blocks every time a chosen event occurs. The events can be: the device button pressed/released, the keyboard key(s) pressed/released, a mouse button pressed/released, an elapsed time, the haptic device touching an object ("Pointer Collision") and monitoring the state (position/time) of a guided movement ("Movement"). The "Movement" option should be used in conjunction with the Switch block with the corresponding option. If desired, it is possible to leave the loop by setting a leaving condition such as a number of occurrences or time. See the user guide pages 8 and 10 for a more detailed description of the parameters.

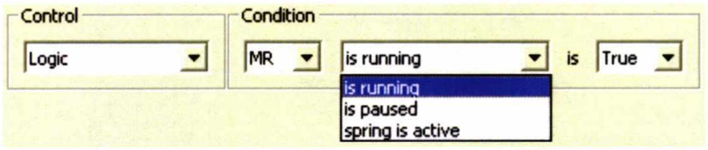

Control: Movement
Movement Sensor
Choose to monitor the movement of: MR through position (position, time)
Leave condition:
[✓ None]
Count 1
Time (after (x) seconds 1
Logic condition

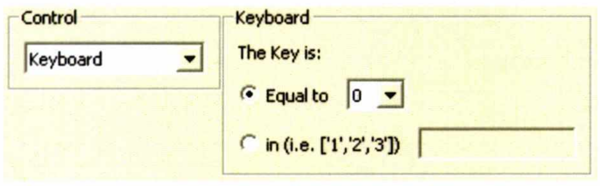

Control
Device Buttons
Device Buttons
Keyboard
Mouse Buttons
Time
Logic
Pointer Collision
Movement
Spring Sensor
Device Button Sensor:
Action: [✓ Pressed] Released Bumped?
Leave condition:
[✓ None]
Count 1
Time (after (x) seconds 1
Logic condition



Switch and **Switch_end:** allows testing if a condition is satisfied or not before executing a set of actions contained between the two blocks. It is used after a Wait For or Everytime block. The Switch block has two (only and always two, even if a sequence is empty) lines departing from it: the upper line tests the "if condition chosen is satisfied" statement while the lower line represent the "if condition chosen not satisfied statement". There are 4 main tests: *Keyboard* - the value of the key pressed; *Logic* - value of some of the parameters of the Guidance_Add and SpringEffect from the Haptic Effect blocks; *Movement sensor* - when used in conjunction with the Everytime block to test the value of the current position or elapsed time and *Compare* - to test if some values are equal. The other tests are not available yet. See the user guide pages 11-13 for a more detailed description of the parameters.


Control
Movement sensor
Movement
MR position in
-0.01, -0.03, 0; 0.01, 0.02, 0


Control
Logic
Condition
MR is running is True
is running
is paused
spring is active


Control
Keyboard
Keyboard
The Key is:
Equal to
0
in (i.e. ['1','2','3'])

B.5 Questionnaire

Tutorial:

1. Was the tutorial easy to understand?

Haptic technology:

1. Did you find the haptic device difficult to use?

The prototyping software:

The blocks icons:

1. Did you like the images used for the blocks?
2. Did the image blocks correspond to the functionalities you expected them to have?
3. Did you find it useful that the image block displays parameters once they are selected?

The bottom panel:

1. Was the bottom panel easy to use to control the parameters?

The interactions within the tool:

1. Were the drag and drop, selection and linking interactions easy to use?
2. Were there some interactions missing that you would like to be available?

Overall use:

1. Was the tool easy to use?
2. Did the tool enable you to prototype and test interactions?
3. Do you think that if you needed to prototype haptic interactions, you would use the tool? Or would you prefer learning the programming languages to develop the interactions?
4. List things you liked about the tool.
5. List things you didn't like about the tool.
6. What improvements do you suggest for the tool?
7. Do you have any other comments?

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

B.6 Debriefing form

Thank you very much for your participation in this research. The HITPROTO tool aims at allowing rapid and easy prototyping of haptic interactions for people with no or little programming skills. Therefore this study was designed to evaluate whether the tool is usable for this.

If you have any queries about this research or would like to ask any further questions, please contact me using the contact details below. As the study is still on-going, please do not talk about the study and the procedures to any other potential participants.

Once again, we would like to thank you for your valuable contribution to this research. Your participation is greatly appreciated.

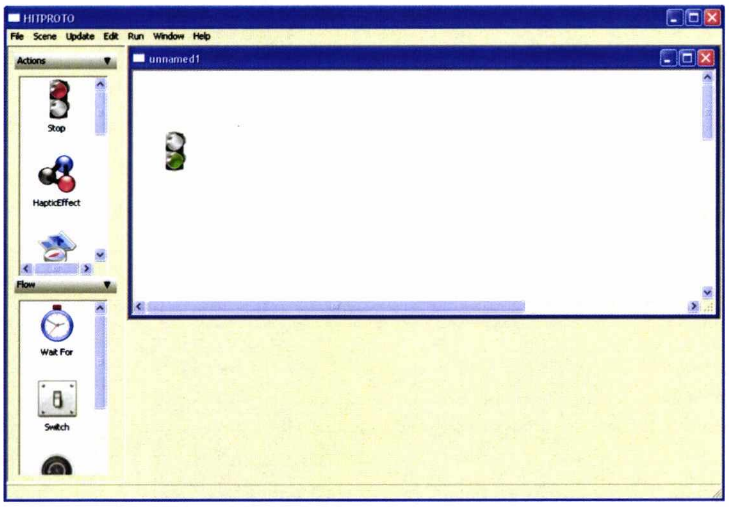
Yours sincerely,
Sabrina Panëels

Researcher contact details: sap28@kent.ac.uk

Appendix C

HITPROTO Pilot Study Tutorial

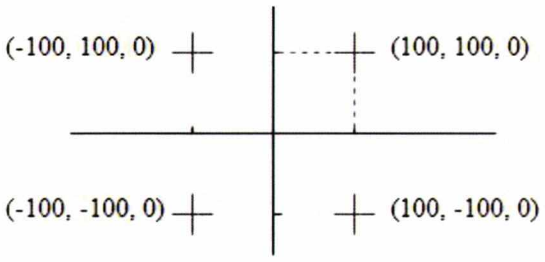
Create a new diagram: In the upper left corner menu, select 'File' - 'New' - 'Template - Diagram' - 'OK'. A new unnamed window opens with a green light, the *Start* shape.



Now let's create three interaction scenarios to see how to use the different block shapes.

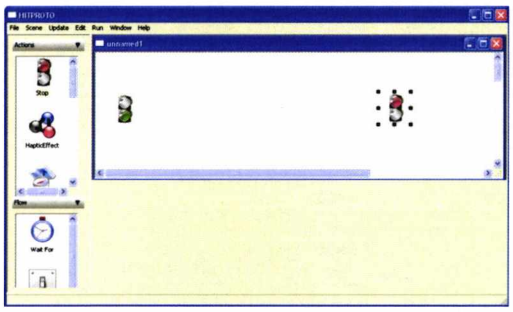
Scenario 1: create and add to the scene some magnetic lines similar to the demo tried earlier. The coordinates of the points are: (100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0). After the keyboard key 'a' has been pressed, remove the magnetic lines.
NB: the scene here refers to the world where the haptic device is used.

Reminder of the coordinates:



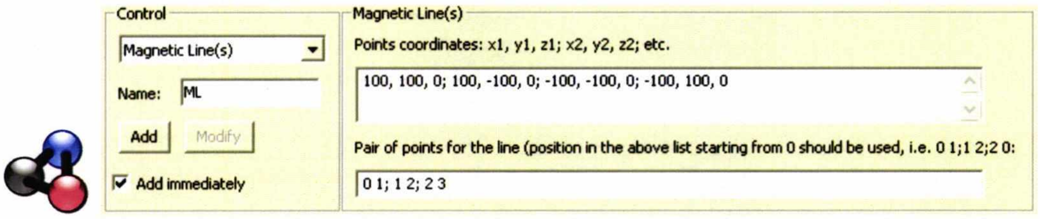
To construct the interaction diagram:

a) First, let's add the compulsory *Stop* shape. The block shapes are located in the left panel. Scroll the Actions shape, select the *Stop* shape and drag'n'drop it towards the right hand side of the window. The shape can be moved around by dragging it with the left mouse button. The rest of the blocks will go between the *Start* and *Stop* shape.

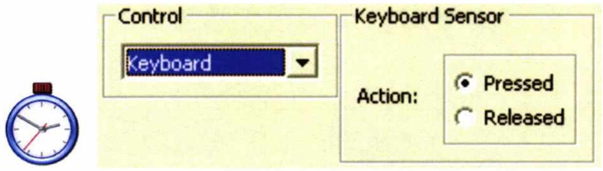


b) Add the rest of the blocks:

- The first stage involves the creation of the magnetic lines. Add the *Haptic Effect* block by drag'n'dropping it on the right side of the *Start* shape. At the bottom panel, you can see the block's parameters displayed. Select the option *Magnetic Line(s)* from the drop down list which enables the creation of magnetic lines. As the haptic effect is an "instance", we need to give it a name to be able to refer to it from other blocks. Call it 'ML' and click the *Add* button. If you are unhappy with the name, click *Modify*, change the name and click *Add* again. Once named, we need to set the point coordinates which gives the points position in 3D space. Please input these coordinates: 100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0 by respecting the format. Next, we need to set the pair of points' positions in the list of coordinates to specify how to connect the lines. The point position in the list starts at number 0. Here we want to construct 3 lines going down, straight and up which translates to point 0-point 1 for line 1, point 1-point 2 for line 2 and point 2-point 3 for the last line. This gives the following pairs: 0 1; 1 2; 2 3. Please input them respecting the format. Also because we want to create and add the instance at the same time, we check the checkbox "Add immediately". It is possible to create an instance but add it after the occurrence of an event, through the *Add_Modify* block.

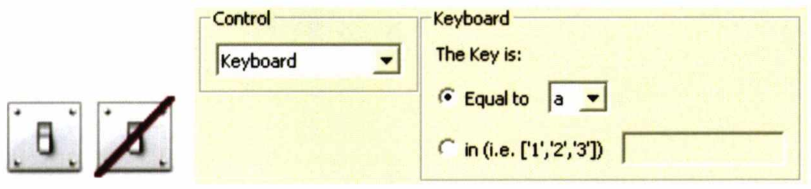

Control
Magnetic Line(s)
Name: ML
Add
Modify
[✓ Add immediately]
Magnetic Line(s)
Points coordinates: x1, y1, z1; x2, y2, z2; etc.
100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0
Pair of points for the line (position in the above list starting from 0 should be used, i.e. 0 1;1 2;2 0:
0 1; 1 2; 2 3

- The second stage, which deals with the removal of the lines, involves the following three blocks: *Trash*, *Wait For* and *Switch* (to test conditions). We could either tune them separately and then find the right order or find the right order first and tune them sequentially. We will follow the latter procedure. The magnetic lines should be removed after the key 'a' is pressed. This means we need to wait for a keyboard pressed event first, then test if it's the key 'a' and finally remove the magnetic lines. Thus the order is: *Wait For*, *Switch* and *Trash*. We thus select the *Wait For* block from the 'Flow' blocks left panel and drag it next to the *Haptic Effect* block. In the bottom panel, select from the drop down list the option Keyboard set to 'pressed'.

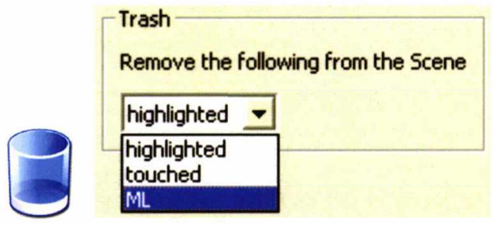

Control
Keyboard
Keyboard Sensor
Action: [✓ Pressed] Released

- Then we drag the *Switch* block next to the *Wait For* block. You will notice that two block shapes are dropped, the *Switch* block and the *Switch_end* block. These blocks allow the testing of whether some conditions are satisfied. Two lines depart from the *Switch* block and connect back to the *Switch_end* block. There should always be two lines, even if there are no actions to do. The upper line tests "if the selected condition is satisfied" while the lower line tests "if the selected condition is not satisfied".

We also choose the *Keyboard* option. There we have the choice of testing the key against a value or a set of values. As we only need to find the key 'a', we choose the first option and we choose the value 'a' from the list. Thus the test will read as: (upper line) if the key value is equal to a, do the following actions; (lower line) if not, do the following actions.

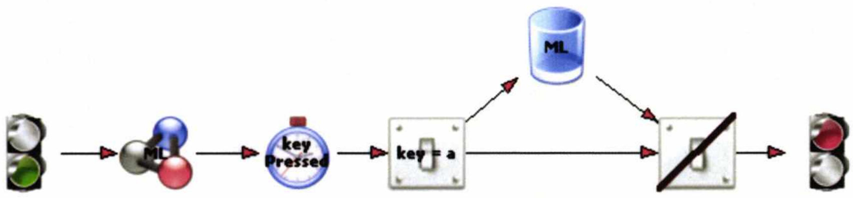

Control
Keyboard
Keyboard
The Key is:
[✓ Equal to] a
in (i.e. ['1','2','3'])

- Lastly, we drag and tune the *Trash* block. As the removal happens if the key 'a' is pressed, we drop the *Trash* block between the *Switch* and *Switch_end* blocks, slightly above them, so that the block will be on the upper line. We just need to select which instance we want to remove, using its name, in this case 'ML'.


Trash
Remove the following from the Scene
highlighted
highlighted
touched
ML

c) If not already done, we link the block shapes in the right order, starting from the *Start* shape until the *Stop* shape. The linking is done by dragging the right mouse button from the centre of the *Start* shape until the centre of the *Haptic Effect* shape, and so on until the *Stop* shape. Let's not forget that two lines depart from the *Switch* shape and that even if nothing happens for one condition, two lines are always needed.

Your diagram should now look like the following:



d) Run the diagram: menu 'Run' option 'Run'. To exit the opened scene, simply press the 'Esc' key. To re-run the diagram, the same procedure needs to be followed each time.

Question: Run the diagram again, what happens if the first key pressed is not 'a'? How can this be solved?

Scenario 2 (also described in the user manual): create a guided movement that is added and started at the device's position after the device's button has been pressed. The path is: (-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0).

a) Open a new diagram like previously.

b) Add the *Stop* block.

c) Identify the needed blocks: guided movements, where the device gets attached to a moving anchor through a spring force and led along a path by it, are achieved using the *Guidance_Add* block to create the guidance instance and the *Guidance_Control* block to control the movement (start, pause, resume or stop it). If the addition is not done immediately (as opposed to the previous scenario), then the block *Add_Modify*, which enables the addition or modification of instances, is needed to add the guidance instance.

   In this case, two interpretations can be made. It is clear that starting the guidance happens after a certain event; however the addition could be done either before or after the event. We will develop the diagram for the latter case.

   Similar to the previous scenario, 'after [event]' refers to waiting for an event to happen and only once, using the *Wait For* block. In this case we need to wait until the device's button is pressed. It is worth noting that usually you need to ensure that the device is attached to the moving anchor, which is done by waiting for the guidance's spring to get active. In this case, as the guidance is added directly at the device's position, the device gets attached automatically and therefore this extra test is not necessary.

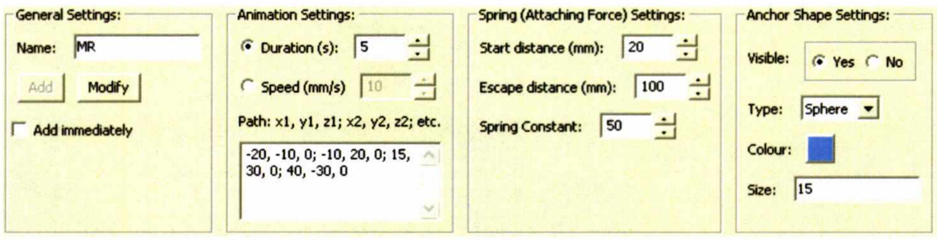
d) Ordering and tuning the parameters: the ordered blocks needed are *Guidance_Add* (to create the instance), *Wait For* the device to be pressed, *Add_Modify* to add the guidance to the scene and lastly *Guidance_Control* to start it.

- *Guidance_Add* block: other than parameters such as the name (input 'MR'), the path (please input the path) or the speed/duration, the guidance contains parameters for the anchor shape and the spring force.

  The parameters for the spring force need to be carefully set for the desired outcome. It includes a constant, a start distance and an escape distance. The start distance is the distance within which the effect works and attracts you. Once attached to the spring, the escape distance is the distance within which you will stay attached to the spring. The escape distance should always be superior or equal to the start distance. The constant determines the intensity of the force.

  For instance, if you want to be able to get away from the spring easily, low values for the constant (50-100) and escape distance (if start distance = 10mm then the escape distance should be 30mm) are needed. Otherwise, if you want to be tightly attached to the spring, then high values for the constant (200) and the escape distance (100mm) are needed.

  As the guidance will be added after the button press, the option "Add immediately" is left unchecked.

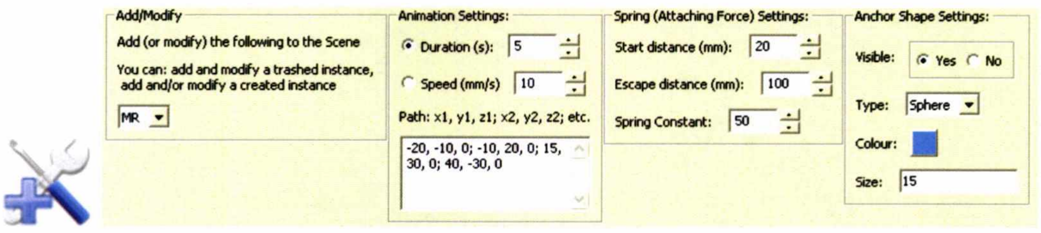

General Settings:
Name: MR
Add | Modify
☐ Add immediately

Animation Settings:
[✓ Duration (s)]: 5
Speed (mm/s): 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 15

- *Wait For* block: we set it to the device buttons option.
- *Add_Modify* block: as we are only adding the guidance, and we do not need to modify the parameters, we just need to select the name of the guidance instance 'MR' we are adding to the scene and leave the other parameters as they are.



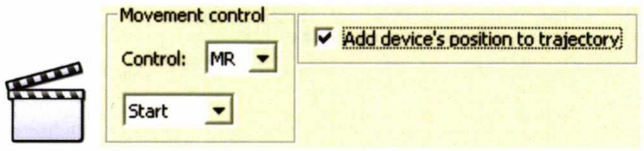
Add/Modify
Add (or modify) the following to the Scene
You can: add and modify a trashed instance, add and/or modify a created instance
MR

Animation Settings:
(● Duration (s): 5
○ Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

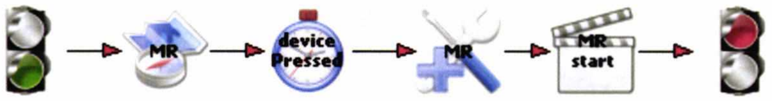
Anchor Shape Settings:
Visible: ● Yes ○ No
Type: Sphere
Colour: 🟦
Size: 15

- *Guidance_Control* block: we add this block to start the guided movement. First, we select the name of the guidance to be controlled, 'MR', then the control we wish to use, in this case *Start*. As we need to start the movement at the device's position, we check the box 'Add device's position to trajectory'.



Movement control
Control: MR
Start
✓ Add device's position to trajectory

e) Link the shapes. Your diagram should now look like this:



f) Run and test the interaction.

Question: Run the interaction again and try to escape the guidance while it is running. What happens? What should be done to prevent that?

Scenario 3: Each time the pointer touches a new object, change the surface's object to be magnetic and change the previously touched object's surface back to be non-magnetic. Also, each time the device's button is pressed, the object with the magnetic surface gets selected.
NB: For this scenario, we need to use a scene which already contains 3D objects. You need not to worry about how to create such a scene. You just need to know that objects from the scene have names such as 'BOX1' that you can use to refer to the object and that you can load a scene by going into the menu 'Scene'.

a) First we need to 'load' the scene. Go into the menu 'Scene', select 'Open X3D Scene' and then choose the scene file called — objectss.x3d' located in the shortcut folder ExperimentData on the desktop and within that folder in the subfolder ExperimentScenes. This scene contains two spheres called 'SPHERE1' and 'SPHERE2', and two boxes called 'BOX1' and 'BOX2'.

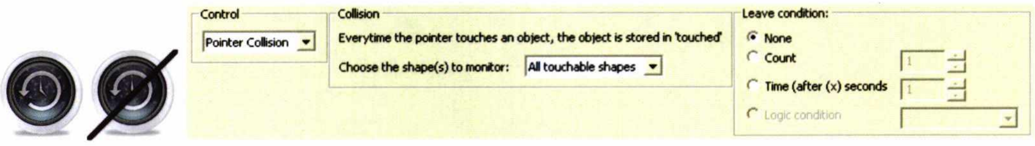
b) Drag'n'drop the *Stop* shape on the right side of the diagram.

c) Identify the needed blocks: from the scenario, the actions required are changing an object surface to be magnetic, changing it back to non-magnetic and 'select' it. Changing a surface to be magnetic is a haptic effect. However, in this particular case, the *Highlight* block needs to be used. It allows the 'highlighting' of a chosen object through a selection of haptic effects. When the point of view is object-oriented, it allows the addition of haptic effects without the concern of the effects' positions (such as for a spring or a magnetic line) by just selecting the object to apply the effect to. Also for now, the range of 'highlighting' effects is limited to a few effects, such as making it magnetic, attractive or surrounding it with a magnetic bounding box. Naturally, removing the highlighting effects is done using the *Unhighlight* block. The selection is done through the *Select* block. These actions happen 'each time' a certain event happens. This is achieved with the *Everytime* block, which is similar to the *Wait For* block as you wait for events to happen, but you wait for them repeatedly.

d) Ordering and tuning the block parameters: there are two separate set of actions. One deals with the highlighting of objects and the other one with the selection of highlighted objects. At this happens in parallel and not sequentially, we will use two sequence lines departing from the *Start shape*. Both lines depend on some events, so both will start with the *Everytime* block. But let's do one line at a time.

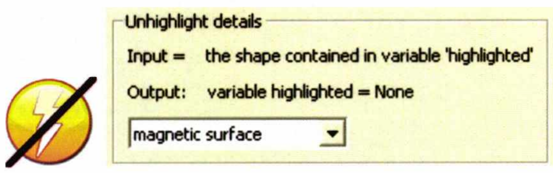
  - *Everytime* block: it constantly waits for events to happen. Before highlighting objects, we need to know whether the haptic device has touched any. After dropping the block next to the *Start* shape, select the option 'Pointer Collision'. This will monitor whether the device touches an object. If it does, the name of the object will be stored in the container 'touched'. Therefore we can make tests or actions on the object without needing to know which one it is exactly each time. In this case, as we are interested in highlighting any object touched, we select the option

"All touchable shapes" from the drop down list. As we want the device 'collisions' to be monitored all the time, we do not need to bother with the leaving conditions. You will also notice that, similarly to the *Switch* block, two shapes are dropped. This is because we need to specify which actions to repeat within these two blocks.



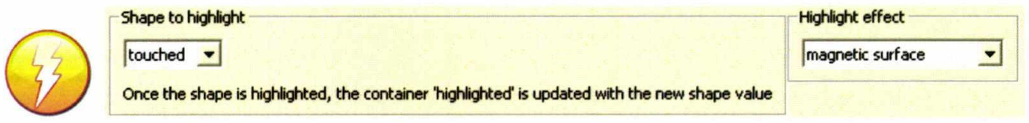
Control
Pointer Collision
Collision
Everytime the pointer touches an object, the object is stored in 'touched'
Choose the shape(s) to monitor: All touchable shapes
Leave condition:
[✓ None]
Count 1
Time (after (x) seconds 1
Logic condition

- *Switch* block: even though the next actions could simply be the highlighting/unhighlighting, we consider the case where the same object is touched twice in a row. Then, we do not need to unhighlight and highlight it again; we can simply leave it as it is. We therefore add a *Switch* block to test whether the object touched stored in 'touched' is the same as the one currently highlighted stored in 'highlighted'. We use the option 'Compare' with the test 'highlighted - Equal to - touched'. If it's the case, nothing happens, so the first line will be empty but will still need to be linked. The second line will contain the highlighting blocks.
- *Unhighlight* block: as the highlighted object is stored in the container 'highlighted' and gets overridden each time a new object is highlighted, we need to do the 'unhighlighting' before we highlight the new touched object. We unhighlight by specifying which effect to remove, here the magnetic surface.

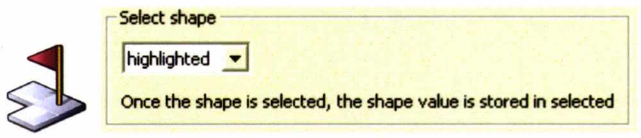


Unhighlight details
Input = the shape contained in variable 'highlighted'
Output: variable highlighted = None
magnetic surface

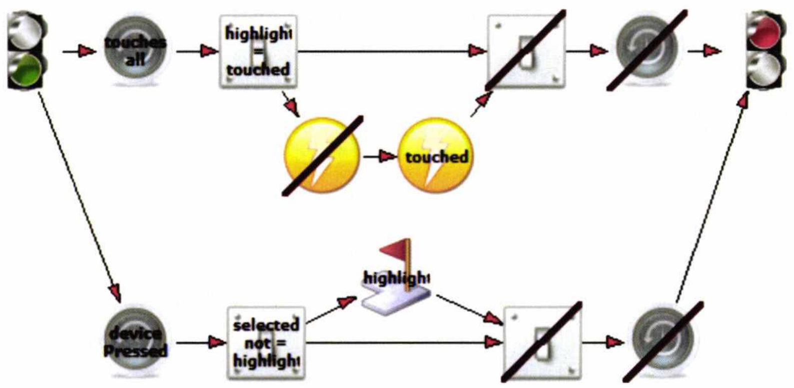
- *Highlight* block: finally we highlight the object by selecting which effect to add, here the magnetic surface and which object to apply it to. As we are monitoring all the objects, the only way to know the touched object is through the 'touched' container, which we select in the object list.


Shape to highlight
touched
Once the shape is highlighted, the container 'highlighted' is updated with the new shape value
Highlight effect
magnetic surface

- Second set of actions: we listen to the device button events with the *Everytime* block. Similar to the highlighting actions, we could just select the current highlighted object, but as it is unnecessary in the case where the object is already selected, we add an extra test with the *Switch* block, where we compare the object selected stored in the container 'selected' with the 'highlighted' object in the respective container. For a bit of change, we choose the 'not equal' test this time. This way the lower line does nothing, while the upper line selects the highlighted object using the Select block.


Select shape
highlighted
Once the shape is selected, the shape value is stored in selected

e) Link the shapes as previously explained and do not forget the empty lines.



f) Run and test the interaction.

Check yourself!

Now, it is your turn to try to complete a scenario on your own. The solution will be provided afterwards. I can help you with this scenario but not for later scenarios during the evaluation phase.

Scenario 4: create and add an interaction where the user will be led along a given path by a visible anchor. It will start after the device is first attached to the anchor (hint: after the collocated spring becomes active). Once it has started, the movement pauses, and resumes after 2 seconds, every time the keyboard has been pressed. Use the following path: (-100, -80, 0; -50, 20, 0; 15, 30, 0; 40, -30, 0).

Scenario 4 - Solution:

First we need to 'translate' the scenario into a scenario that makes it clearer which blocks should be used.

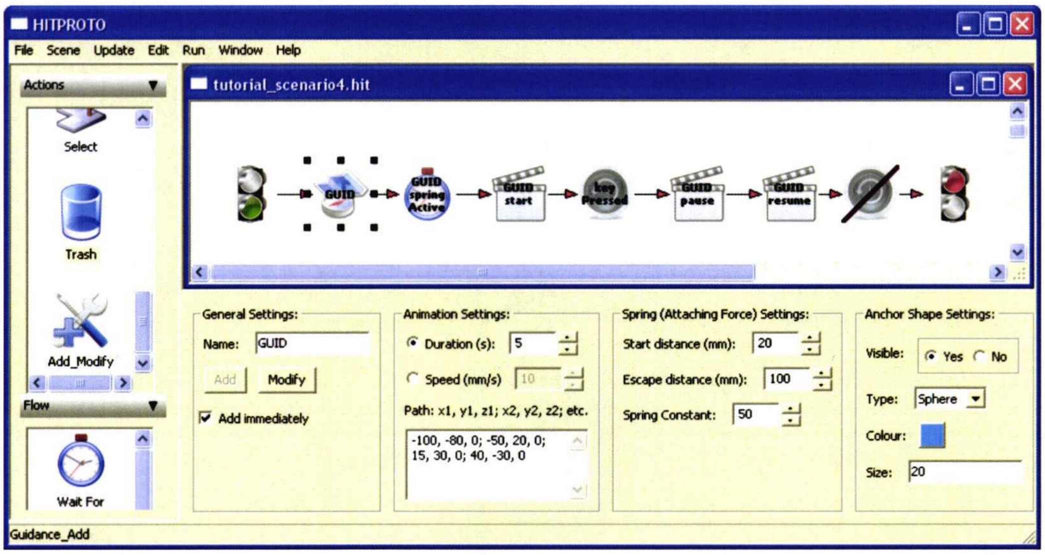
- User will be led along a given path by a visible anchor: it refers to the guidance interaction blocks.
- User/device gets attached to the anchor: this refers to the device getting 'snapped' by the anchor included in the guidance interaction. We know if the device is attached when the guidance's spring becomes active.
- After the user gets attached []: it could be either a reference to a *Wait For* or an *Everytime* block, but because starting the guidance is done only once, this means the *Wait For* block should be used.
- The movement pauses and resumes every time: both the *Guidance_Control* block and *Everytime* block will be needed.
- Every time the keyboard is pressed: this means the *Everytime* block should be used, tuned to listening to keyboard events.

The scenario translates into:
Create and add immediately a guidance interaction. Wait for the guidance's spring to be active, and then start the guidance. Once it is running, every time the keyboard is pressed, then pause the guidance, and resume after 2 seconds.

The diagram construction breaks down into:

1. Adding the *Stop* block.
2. Adding the *Guidance_Add* block, giving it a name (i.e. 'GUID') and checking the option 'Add immediately'. Setting the path to the given values and the duration to 5 seconds for instance.
3. Adding the *Wait For* block, select the spring sensor option and the guidance's name to monitor.
4. Add the *Guidance_Control* block set to start.
5. Add the *Everytime* block, set to 'Keyboard'.

6. Add the *Guidance_Control* block set to pause for the given guidance instance and with the option 'do nothing'.
7. Add the *Guidance_Control* block to resume the guidance, set to 2 seconds.
8. Then we link the shapes.



General Settings:
Name: GUID
Add | Modify
[✓ Add immediately]

Animation Settings:
[✓ Duration (s):] 5
Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-100, -80, 0; -50, 20, 0;
15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 20

Any questions?

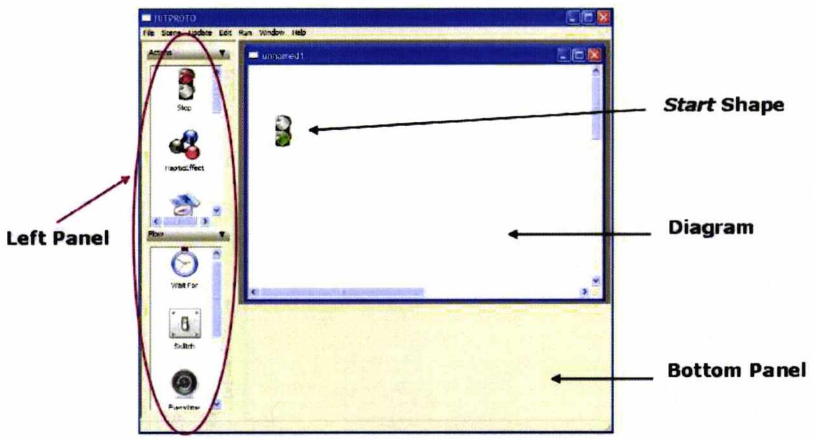
Well Done! You can keep this tutorial when performing the tasks. The blocks presented above are summarized in the blocks summary sheet and described in more details in the user manual, which the blocks summary sheet also refers to.

Now let's have a break!

Appendix D

Experiment New Materials: Modified Tutorial and New Visual Summary

Create a new diagram: In the upper left corner menu, select 'File' - 'New' - 'Template - Diagram' - 'OK'. A new unnamed window opens with a green light, the *Start* shape.

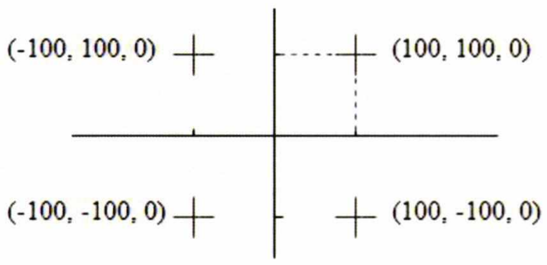


Now let's create three interaction scenarios to see how to use the different block shapes.

Scenario 1: create and add to the scene some magnetic lines similar to the demo tried earlier. The coordinates of the points are: (100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0). After the keyboard key 'a' has been pressed, remove the magnetic lines.

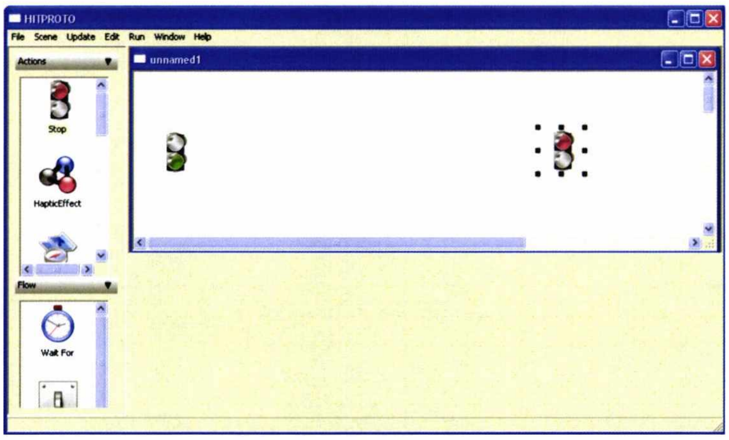
NB: the scene here refers to the world where the haptic device is used.

Reminder of the coordinates:



To construct the interaction diagram:

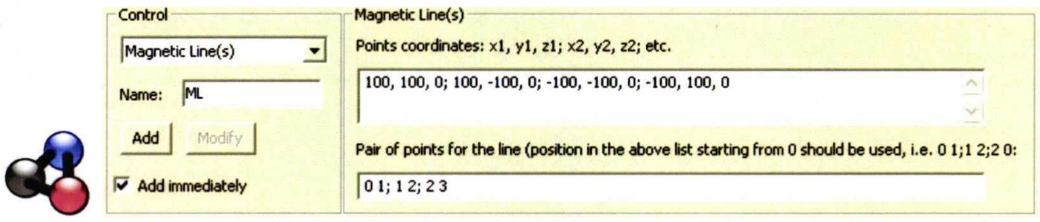
a) First, let's add the compulsory *Stop* shape. The block shapes are located in the left panel. Scroll the Actions shape, select the *Stop* shape and drag'n'drop it towards the right hand side of the window. The shape can be moved around by dragging it with the left mouse button. The rest of the blocks will go between the *Start* and *Stop* shape.



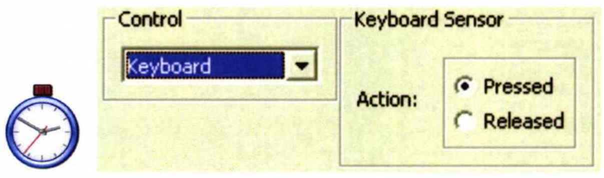
b) Add the rest of the blocks:

- The first stage involves the creation of the magnetic lines. Add the *Haptic Effect* block by drag'n'dropping it on the right side of the *Start* shape. At the bottom panel, you can see the block's parameters displayed. Select the option *Magnetic Line(s)* from the drop down list which enables the creation of magnetic lines. As the haptic effect is an "instance", we need to give it a name to be able to refer to it from other blocks. Call it 'ML' and click the *Add* button. If you are unhappy with the name, click

Modify, change the name and click *Add* again. Once named, we need to set the point coordinates which gives the points position in 3D space. Please input these coordinates: 100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0 by respecting the format. Next, we need to set the pair of points' positions in the list of coordinates to specify how to connect the lines. The point position in the list starts at number 0. Here we want to construct 3 lines going down, straight and up which translates to point 0-point 1 for line 1, point 1-point 2 for line 2 and point 2-point 3 for the last line. This gives the following pairs: 0 1; 1 2; 2 3. Please input them respecting the format. Also because we want to create and add the instance at the same time, we check the checkbox "Add immediately". It is possible to create an instance but add it after the occurrence of an event, through the *Add_Modify* block.

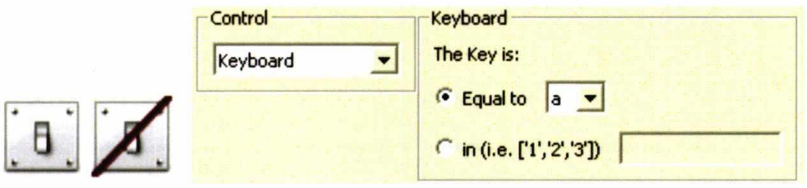

Control
Magnetic Line(s)
Name: ML
Add Modify
✓ Add immediately
Magnetic Line(s)
Points coordinates: x1, y1, z1; x2, y2, z2; etc.
100, 100, 0; 100, -100, 0; -100, -100, 0; -100, 100, 0
Pair of points for the line (position in the above list starting from 0 should be used, i.e. 0 1;1 2;2 0:
0 1; 1 2; 2 3

- The second stage, which deals with the removal of the lines, involves the following three blocks: *Trash*, *Wait For* and *Switch* (to test conditions). We could either tune them separately and then find the right order or find the right order first and tune them sequentially. We will follow the latter procedure. The magnetic lines should be removed after the key 'a' is pressed. This means we need to wait for a keyboard pressed event first, then test if it's the key 'a' and finally remove the magnetic lines. Thus the order is: *Wait For*, *Switch* and *Trash.* We thus select the *Wait For* block from the 'Flow' blocks left panel and drag it next to the *Haptic Effect* block. In the bottom panel, select from the drop down list the option Keyboard set to 'pressed'.


Control
Keyboard
Keyboard Sensor
Action: [✓ Pressed] Released

- Then we drag the *Switch* block next to the *Wait For* block. You will notice that two block shapes are dropped, the *Switch* block and the *Switch_end* block. These blocks allow the testing of whether some conditions are satisfied. Two lines depart from the *Switch* block and connect back to the *Switch_end* block. There should always be two lines, even if there are no actions to do. The upper line tests "if the selected condition is satisfied" while the lower line tests "if the selected condition is not satisfied".

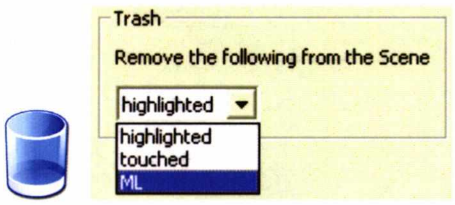
  We also choose the *Keyboard* option. There we have the choice of testing the key against a value or a set of values. As we only need to find the key 'a', we choose the first option and we choose the value 'a' from the list. Thus the test will read as: (upper line) if the key value is equal to a, do the following actions; (lower line) if not, do the following actions.



Control
Keyboard

Keyboard
The Key is:
(•) Equal to a
( ) in (i.e. ['1','2','3'])

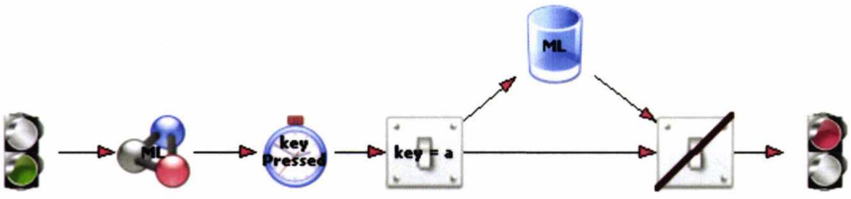
- Lastly, we drag and tune the *Trash* block. As the removal happens if the key 'a' is pressed, we drop the *Trash* block between the *Switch* and *Switch_end* blocks, slightly above them, so that the block will be on the upper line. We just need to select which instance we want to remove, using its name, in this case 'ML'.



Trash
Remove the following from the Scene
highlighted
highlighted
touched
ML

c) If not already done, we link the block shapes in the right order, starting from the *Start* shape until the *Stop* shape. The linking is done by dragging the right mouse button from the centre of the *Start* shape until the centre of the *Haptic Effect* shape, and so on until the *Stop* shape. Let's not forget that two lines depart from the *Switch* shape and that even if nothing happens for one condition, two lines are always needed.

   Your diagram should now look like the following:



d) Run the diagram: menu 'Run' option 'Run'. To exit the opened scene, simply press the 'Esc' key. To re-run the diagram, the same procedure needs to be followed each time.

Question: Run the diagram again, what happens if the first key pressed is not 'a'? How can this be solved?

Answer: Nothing because we wait for the keyboard to be pressed only once. By changing the *Wait For* block for the *Everytime* block. The *Everytime* block constantly waits for events to happen. Instead of waiting for the first key to be pressed, it monitors each key that has been pressed. So each time a key is pressed, we can do something. After adding it, you will also notice that, similarly to the *Switch* block, two shapes are dropped. This is because we need to specify which actions to repeat within these two blocks, which show the beginning and end of the repetition. Similarly to *Wait For* we select the Keyboard option.

Scenario 2 (also described in the user manual): create a guided movement that is added and started at the device's position after the device's button has been pressed. The path is: (-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0).

a) Open a new diagram like previously.

b) Add the *Stop* block.

c) Identify the needed blocks: guided movements, where the device gets attached to a moving anchor through a spring force and led along a path by it, are achieved using the *Guidance_Add* block to create the guidance instance and the *Guidance_Control* block to control the movement (start, pause, resume or stop it). If the addition is not done immediately (as opposed to the previous scenario), then the block *Add_Modify*, which enables the addition or modification of instances, is needed to add the guidance instance.

In this case, two interpretations can be made. It is clear that starting the guidance happens after a certain event; however the addition could be done either before or after the event. We will develop the diagram for the latter case.

Similar to the previous scenario, 'after [event]' refers to waiting for an event to happen and only once, using the *Wait For* block. In this case we need to wait until the device's button is pressed. It is worth noting that usually you need to ensure that the device is attached to the moving anchor, which is done by waiting for the guidance's spring to get active. In this case, as the guidance is added directly at the device's position, the device gets attached automatically and therefore this extra test is not necessary.

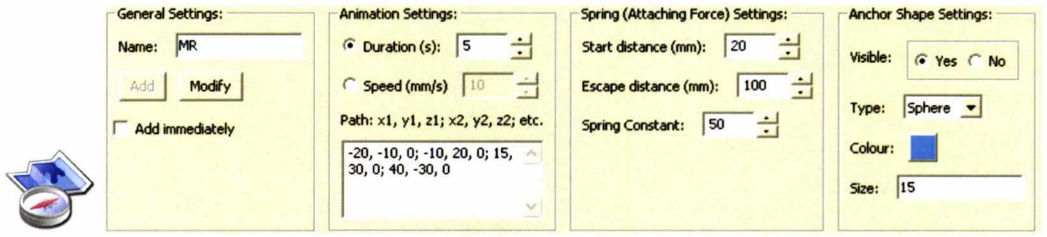
d) Ordering and tuning the parameters: the ordered blocks needed are *Guidance_Add* (to create the instance), *Wait For* the device to be pressed, *Add_Modify* to add the guidance to the scene and lastly *Guidance_Control* to start it.

   - *Guidance_Add* block: other than parameters such as the name (input 'MR'), the path (please input the path) or the speed/duration, the guidance contains parameters for the anchor shape and the spring force.

     The parameters for the spring force need to be carefully set for the desired outcome. It includes a constant, a start distance and an escape distance. The start distance is the distance within which the effect works and attracts you. Once attached to the spring, the escape distance is the distance within which you will stay attached to the spring. The escape distance should always be superior or equal to the start distance. The constant determines the intensity of the force.

     For instance, if you want to be able to get away from the spring easily, low values for the constant (50-100) and escape distance (if start distance = 10mm then the escape distance should be 30mm) are needed. Otherwise, if you want to be tightly attached to the spring, then high values for the constant (200) and the escape distance (100mm) are needed.

     As the guidance will be added after the button press, the option "Add immediately" is left unchecked.

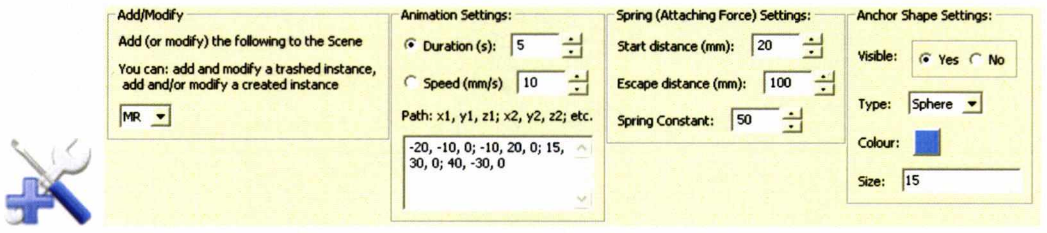

General Settings:
Name: MR
Add | Modify
☐ Add immediately

Animation Settings:
[✓ Duration (s):] 5
Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 15

- *Wait For* block: we set it to the device buttons option.
- *Add_Modify* block: as we are only adding the guidance, and we do not need to modify the parameters, we just need to select the name of the guidance instance 'MR' we are adding to the scene and leave the other parameters as they are.

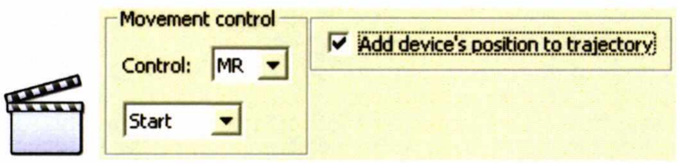

Add/Modify
Add (or modify) the following to the Scene
You can: add and modify a trashed instance, add and/or modify a created instance
MR

Animation Settings:
[✓ Duration (s):] 5
Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

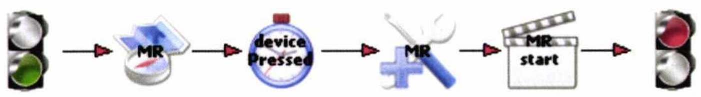
Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 15

- *Guidance_Control* block: we add this block to start the guided movement. First, we select the name of the guidance to be controlled, 'MR', then the control we wish to use, in this case *Start*. As we need to start the movement at the device's position, we check the box 'Add device's position to trajectory'.


Movement control
Control: MR
Start
☑ Add device's position to trajectory

e) Link the shapes. Your diagram should now look like this:



f) Run and test the interaction.

Question: Run the interaction again and try to escape the guidance while it is running. What happens? What should be done to prevent that?

Check yourself!

Now, it is your turn to try to complete a scenario on your own. The solution will be provided afterwards. I can help you with this scenario but not for later scenarios during the evaluation phase.

Scenario 4: create and add an interaction where the user will be led along a given path by a visible anchor. It will start after the device is first attached to the anchor (hint: after the collocated spring becomes active). Once it has started, the movement pauses, and resumes after 2 seconds, every time the keyboard has been pressed. Use the following path: (-100, -80, 0; -50, 20, 0; 15, 30, 0; 40, -30, 0).

Scenario 4 - Solution:

First we need to 'translate' the scenario into a scenario that makes it clearer which blocks should be used.

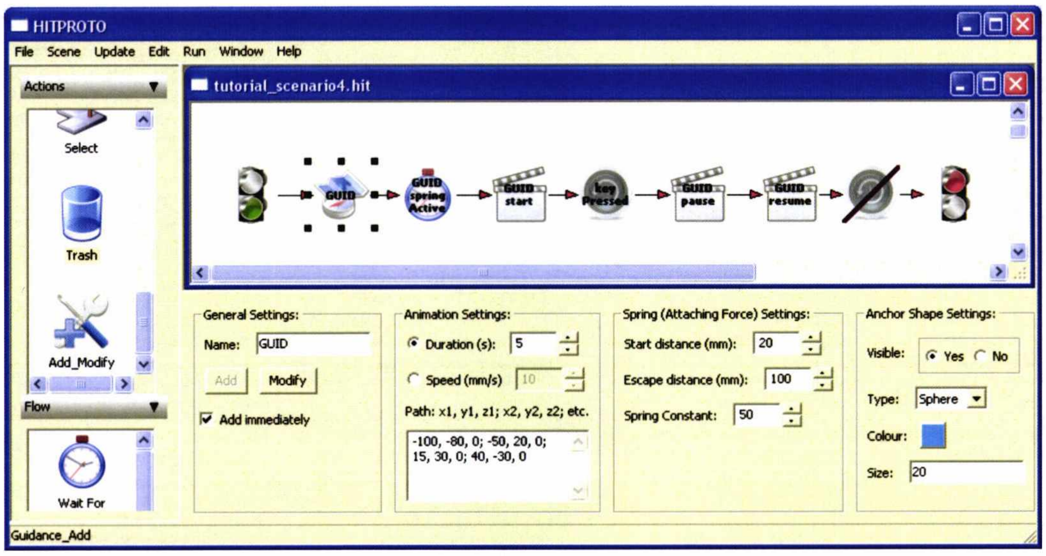
- "The user will be led along a given path by a visible anchor": it refers to the guidance interaction blocks.
- The user/device gets "attached to the anchor": this refers to the device getting 'snapped' by the anchor included in the guidance interaction. We know if the device is attached when the guidance's spring becomes active.
- "**After** the device is first attached": it could be either a reference to a *Wait For* or an *Everytime* block, but because starting the guidance is done only once, this means the *Wait For* block should be used.
- "The movement pauses, and resumes after 2 seconds": this means two *Guidance_Control* blocks will be needed, one set to pause and one set to resume.
- "Every time the keyboard has been pressed": this means the *Everytime* block should be used, tuned to listening to keyboard events.

The scenario translates into:
Create and add immediately a guidance interaction. Wait for the guidance's spring to be active, and then start the guidance. Once it is running, every time the keyboard is pressed, then pause the guidance, and resume after 2 seconds.

The diagram construction breaks down into:

1. Adding the *Stop* block.
2. Adding the *Guidance_Add* block, giving it a name (i.e. 'GUID') and checking the option 'Add immediately'. Setting the path to the given values and the duration to 5 seconds for instance.
3. Adding the *Wait For* block, select the spring sensor option and the guidance's name to monitor.
4. Add the *Guidance_Control* block set to start.
5. Add the *Everytime* block, set to 'Keyboard'.

6. Add the *Guidance_Control* block set to pause for the given guidance instance and with the option 'do nothing'.
7. Add the *Guidance_Control* block to resume the guidance, set to 2 seconds.
8. Then we link the shapes.
9. Finally, test the scenario with the 'Run - Run' commands.



General Settings:
Name: GUID
[✓ Add immediately]

Animation Settings:
[✓ Duration (s): 5] Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-100, -80, 0; -50, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: [✓ Yes] No
Type: Sphere
Colour: 🟦
Size: 20

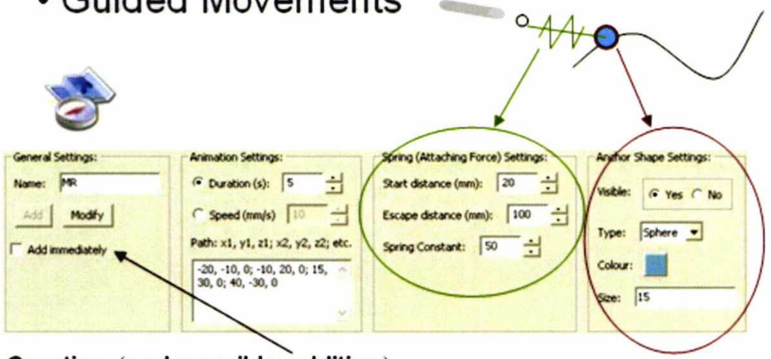
Any questions?

Well Done! You can keep this tutorial when performing the tasks. The blocks presented above are summarized in a visual summary but also in the blocks summary sheet and described in more details in the user manual, which the blocks summary sheet also refers to.

Now let's have a break!

Visual Blocks Summary

- Guided Movements



General Settings:
Name: MR
Add | Modify
☐ Add immediately

Animation Settings:
(●) Duration (s): 5
( ) Speed (mm/s) 10
Path: x1, y1, z1; x2, y2, z2; etc.
-20, -10, 0; -10, 20, 0; 15, 30, 0; 40, -30, 0

Spring (Attaching Force) Settings:
Start distance (mm): 20
Escape distance (mm): 100
Spring Constant: 50

Anchor Shape Settings:
Visible: (●) Yes ( ) No
Type: Sphere
Colour: 🟦
Size: 15

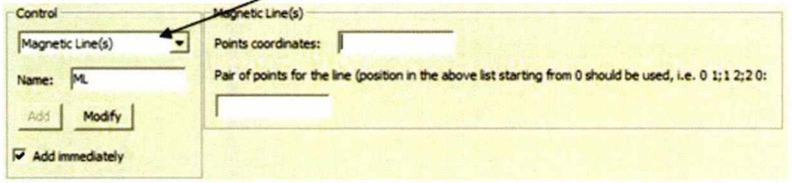
Creation (and possible addition)

Addition and/or Modification of parameters

Control over the movement: starting, pausing, resuming or stopping

Removal

- Haptic Effects: Spring, Magnetic Lines, PositionFunctionEffect



Control
Magnetic Line(s)
Name: ML
Add | Modify
☑ Add immediately

Magnetic Line(s)
Points coordinates:
Pair of points for the line (position in the above list starting from 0 should be used, i.e. 0 1;1 2;2 0:

Creation (and possible addition)

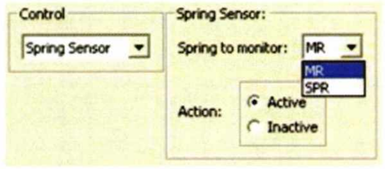
Addition and/or Modification of parameters

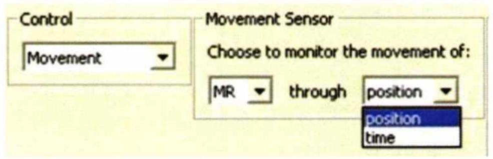
Removal

- Actions dependent on something happening

Something = device button press, mouse button press, keyboard press, spring becoming active (Spring Sensor), device touching objects, monitoring the guided movement (Movement) etc.

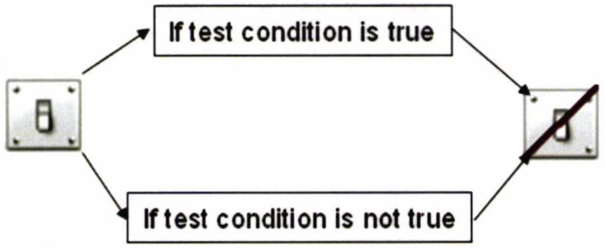
Happening once ***OR*** **Happening repeatedly**

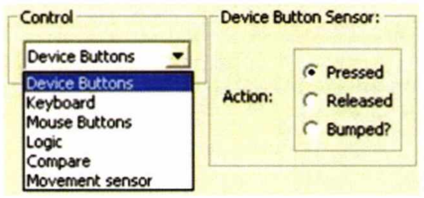

Control: Spring Sensor
Spring Sensor:
Spring to monitor: MR (MR, SPR)
Action: (•) Active ( ) Inactive


Control: Movement
Movement Sensor
Choose to monitor the movement of: MR through position (position, time)

- Tests on what is happening

Tests = which key has been pressed, guided movement paused (Logic), spring active (Logic), guided movement passing by some given positions (Movement sensor), etc.




Control: Device Buttons (Device Buttons, Keyboard, Mouse Buttons, Logic, Compare, Movement sensor)
Device Button Sensor:
Action: (•) Pressed ( ) Released ( ) Bumped?

Appendix E

Experiment Raw Data

E.1 General Participant Data

	Gender	Age	Study Topic
1	F	29	Development studies
2	F	27	Microbiology
3	F	27	Anthropology
4	M	27	Anthropology
5	M	27	Actuarial Science
6	F	22	Psychology
7	F	22	Archeology
8	F	23	Science, communication and society
9	M	24	Visual Effects

E.2 Times and Help

Successful Completion Times (in minutes) & SUS Score

	Task 0	Task 1	Task 2	Task 3	Task 4	SUS
1	31	8	16	23		17.5
2	17	18	6	18	18	77.5
3	36	25	19	17	21	92.5
4	26	19	12	14	14	67.5
5	30	3				50
6	30	15	16	9	33	65
7	22	7	18	23	18	70
8	13	10	10	19	22	85
9	35	16	17	16	36	77.5

Type of help for each task [1]

	Task 0	Task 1	Task 2	Task 3	Task 4
1	S+MaH	S	S+MaH	S+MaH	F
2	S	S+MiH	MiE-H	S+MiH	S
3	S+MaH	S	S+MaH	S+MiH	S+MaH
4	S+MaH	S	MiE-H	MiE-MiH	S
5	S+MaH	S	F	F	!A
6	S+MiH	S+MiH	MiE-H	S+MiH	S+MaH
7	S+MaH	S	S+MiH	S+MaH	S+MaH
8	S	S	MiE-H	S+MiH	S
9	S	S	MiE-H	MiE-H	S+MaH

[1] List of abbreviations:
(S) Success with no help
(S+MiH) Success with minor help
(S+MaH) Success with major help
(MiE-H) Minor errors with no help
(MiE+MiH) Minor errors with minor help
(F) Failure
(!A) Not attempted

E.3 Questionnaire: Summary of Answers

Tutorial:

1. Was the tutorial easy to understand?

"I found it very abstract, especially before doing the actual exercises. For someone not familiar with the language, it's difficult to understand."
"Yes quite easy."
"I think that the terminology could be made easier. Get rid of jargon!"
"Yes but some new terms (could everyday metaphors make it easier)."
"Yes."
"It was okay, but it became more clear once you had started to use the program."
"It was relatively easy to understand, but there was a lot to remember in a short space of time."
"Quite easy but sometimes a bit too fast."
"Not really (the look), video tutorial are easy."

Haptic technology:

1. Did you find using the haptic device difficult to use?

"It would have been easier with a higher chair. It was hard to see properly."
"No, very easy and straightforward."
"No."
"No."
"Initially, you must have understanding of basic programming languages."
"The device itself was easy to use after a little practice and I got used to it."
"No."
"At first but it was easier once I got the hang of it."
"Not really, a lot of logic."

The prototyping software:

THE BLOCKS ICONS:

1. Did you like the images used for the blocks?

"They are ok."
"Yes, not very 'special', but clear and easy to understand."
"Yes! Very helpful! Add an ear to 'wait for' icon." (She wanted ears to emphasize they are listeners and listen to events)
"Yes."

"No: too big icons, better interface could be good, if condition(s) should be independent of location within the diagram."
"Yes."
"Yes, they gave a hint or reminder of what each one was meant to be used for."
"Yes."
"Can be refined."

2. Did the image blocks correspond to the functionalities you expected them to have?
"I find a lot of the functions are hidden. I'm not very clear still what each of them do."
"Yes, mostly."
"Yes."
"Not really, but once learned they were easy to recognise."
"Somewhat."
"Yes, although the everytime clock could be a little clearer." (she wanted the number to appear to show better a clock)
"Yes, and it made understanding their functions a little easier than if they had been represented in another way."
"Yes, maybe the map and compass was a bit confusing until it was explained." (image not quite right, but she couldn't think of anything else that would be better.)
"Not entirely, some of them are confusing."

3. Did you find it useful that the image block displays parameters once they are selected?
"I don't know."
"Yes, very useful to know what they're for."
"Yes."
"Very."
"Certainly."
"Yes it help prompt me what to do."
"Yes."
"Yes."
"Yes, very much."

THE BOTTOM PANEL:

1. Was the bottom panel easy to use to control the parameters?

"Yes. Once you find what you need."
"Yes, quite simple."
"Has a medium difficulty."
"Yes, except when it had parameters from selected objects in a closed window."
"Yes."
"Yes."
"Yes."
"Yes."
"Yes, it's excellent."

THE INTERACTIONS WITHIN THE TOOL:

1. Were the drag and drop, selection and linking interactions easy to use?

"Yes."
"Yes."
"Yes!!"
"Yes."
"Yes."
"Yes - very."
"Yes."
"Very easy."
"Yes they were, but did not like it when we had to delete it, instead of simply disconnecting arrows." (he would have preferred to move away an arrow for it to get deleted, instead of selecting it, then deleting it.)

2. Were there some interactions missing that you would like to be available?

"Not that I can think of."
"Nothing I can think of."
"When you put the mouse over an icon -¿ a few hints with what it does."
"Not yet."
"Instead of a program crash, a compilation would be better before run. BlueJ interface may be an example."
"No."

"No."
"No."
"None."

OVERALL USE:

1. Was the tool easy to use?

"I don't know. Similar to other software I guess."
"After some practise and searching quite easy."
"If I had more time to familiarize myself with it, I think it would be very easy to use."
"Took a while to get used to."
"Yes, after few practice sessions."
"Yes."
"It was easy to use, especially as I have little experience with this type of computer program."
"Yes."
"Very, the drag and drop is the best function."

2. Did the tool enable you to prototype and test interactions?

"I don't know. I just followed the instructions to create a square or moving balls."
"Yes, in ways." (commented that testing not always easy as sometimes it would not work but she didn't know why)
"Yes."
"Trial and error? - yes."
"Definitely, as long as no other interfaces available."
"Yes."
"Yes."
"Yes."
"Yes it did."

3. Do you think that if you needed to prototype haptic interactions, you would use the tool? Or would you prefer learning the programming languages to develop the interactions?

"I think I would get a computing person to do it." (After asking which between the two, she answered she would rather use the tool)
"I would use the tool, easier to understand than programming languages."

"Yes; it is marvellous, I like it (the tool), not learning languages."
"I'd use the tool."
"Using interface is always preferable. But logic statements are as complicated."
"I would rather use the tool than programming language."
"I would rather use the tool as it sounds far less complicated and time consuming."
"Definitely use the tool because I found it easy!"
"Use this tool."

4. List things you liked about the tool.
"There isn't anything in particular that I liked."
"Simple symbols/icons. Easy to understand"
"It's easy for a non-programmer to actually program. It's a relatively easy interface. The fact that it is diagram-based."
"Intuitive. Could run even when task unfinished, useful to make sure you are on the right lines."
"Bottom panel, flow-chart approach."
"Simple layout."
"It made developing/creating things that looked complicated relatively simple. The layout of the tool made it easier to access."
"It was very simple - no need for any programming knowledge."
"Drag and drop functions, connecting the nodes."

5. List things you didn't like about the tool.
"There isn't anything in particular that I disliked."
"Don't know."
"The lack of description when you put the mouse over the icons. I was not always sure about the order of connecting icons."
"Could get messy - grid to keep object in place might be useful (?)." (he wanted a snap-to-grid function).
"Some terms are confusing (and hidden)."
"Lots of things to remember."
"There was nothing I didn't like, although the short space of time meant that I didn't have time to fully understand every element of the tool."
"Some of the 'control' items in the 'Everytime' part were a bit hard to understand e.g 'pointer collision' 'logic'."

"Cannot zoom out, the start node is always stuck in the beginning, icons."

6. What improvements do you suggest for the tool?

"Short descriptions on what each item does and how you need to proceed to get it to do a certain action."
"Maybe a section where you can see what you have created in 'real time', while creating". (Basically showing the result in run-time instead of doing Run-Run each time).
"See 5." (Same as the things she didn't like)
"As above." (the snap-to-grid option).
"Size of icons, error checking/compiling."
No comment.
No comment.
"I can't think of any the only times my things didn't work were when I didn't remember what I had been shown." (maybe an error checking would be useful, or text added saying don't forget to specify the name of the object you want to control)
"Changing the look by modifying the name of the icons, adding a small description once you hover the mouse on the icon to show what it does."

7. Do you have any other comments?

"No."
"Nice tool!"
"No."
"Not yet."
"Look at the next best available software related to this."
"More time to play around and experiment with the tool and program before starting the task."
"No."
"No."
"No."