Prototyping 3D Haptic Data Visualizations

Sabrina A. Panëels[1], Panagiotis D. Ritsos[1,*], Peter J. Rodgers[1], Jonathan C. Roberts[1,**],

[a]CEA, LIST, Sensory and Ambient Interfaces Laboratory, France
[b]School of Computer Science, Bangor University, UK
[c]School of Computing, University of Kent, UK

Abstract

Haptic devices are becoming more widely used as hardware becomes available and the cost of both low and high fidelity haptic devices decreases. One of the application areas of haptics is Haptic Data Visualization (HDV). HDV provides functionality by which users can feel and touch data. Blind and partially sighted users can benefit from HDV, as it helps them manipulate and understand information. However, developing any three-dimensional haptic world is difficult, time-consuming and requires skilled programmers. Therefore, systems that enable haptic worlds to be rapidly developed in a simple environment could enable non-computer skilled users to create haptic 3D interactions. In this article we present HITPROTO: a system that enables users, such as mentors or support workers, to quickly create haptic interactions (with an emphasis on HDVs) through a visual programming interface. We describe HITPROTO and include details of the design and implementation. We present the results of a detailed study using postgraduate students as potential mentors, which provides evidence of the usability of HITPROTO. We also present a pilot study of HITPROTO with a blind user. It can be difficult to create prototyping tools and support 3D interactions, therefore we present a detailed list of 'lessons learnt' that provides a set of guidelines for developers of other 3D haptic prototyping tools.

Keywords: Haptic Data Visualization, Haptics, Haptification, Rapid Prototyping, Haptic Interaction Techniques

1. Introduction

The use of haptic devices is increasing and becoming part of our every-day life. Not only are haptic technologies being used in the home-games market, but they are common place on mobile phones, and are also gaining presence in control systems in the interfaces of commonly used items, such as motor cars. They are especially useful for adding touch to three-dimensional environments.

In particular, one developing application area is the use of haptic devices to display data. This general area is named 'haptic data visualization' and it uses dynamic computer-operated haptic devices to allow users to feel a representation of some data in a three-dimensional world. There are two motivations [1], either to incorporate haptic techniques into data visualization to provide a holistic view of the data and to utilise the haptic modality alongside visual, or to display the information for visually impaired humans. In this article we focus on the latter: to interactively display the data for blind or partially sighted users.

Most of the current haptic visualization practices for visually impaired humans fall into two overarching camps: either the mentors create tactile materials, for instance, printing the graphic onto a special paper that swells (raises) the black ink on heating the paper, or programmers create one-off bespoke demonstrations that are specifically developed to demonstrate a particular representation. Although the former method is static, it is cheap and affords reuse and accessibility. While the latter method enables dynamic interaction and the ability to change the visual depictions, the software development process is long and complex: these systems are developed by skilled programmers and require expert knowledge of the haptic device. Consequently, what is required is a system that can generate haptic data visualizations and can be operated by non-specialist operators. The end result should be a haptic representation that depicts the data and allows the user to understand the information through the medium of touch and force feedback.

Complexity is not only evident in haptic data visualization. Even with the growth and widespread use of haptic devices, the creation of three-dimensional haptic environments is still a time-consuming process. To create a three-dimensional haptic world, a skilled programmer needs to write suitable code that describes the three-dimensional scene and how the haptic device will react and activate. While it is possible to easily convert a three-dimensional scene-graph into a solid haptic world, it is generally difficult to add interaction and develop complex haptic worlds with this process; these quick conversions remain predominantly for static worlds.

Our vision is to enable end-users such as mentors or teachers to quickly create haptic data visualizations that can be used by blind or partially sighted users. The created visualizations can then be analysed and used to understand the underlying data. This requires a rethink of current practices. The HITPROTO

*Corresponding author
**Principal corresponding author
Email addresses: a.email@domain.com (Person A), b.email@domain.com (Person B), c.email@domain.com (Person C), d.email@domain.com (Person D)

system is intended to realise this vision. HITPROTO enables three-dimensional haptic worlds to be *prototyped* quickly and allows users to *interactively* explore these three-dimensional worlds, ultimately aimed at allowing mentors and support workers to create Haptic Data Visualizations.

The term prototyping in Software Engineering (SE) represents the rapid development of a software solution. It provides a result that can act as a 'proof of concept'. The idea is that the developer quickly creates something that contains the key functionality of the end implementation. Prototypes enable developers to trial different scenarios and experiment with the outcome, before investing lots of development time into a producing fully fledged system. Either these prototypes are discarded after they have been developed, and new software is created for the final product, or the product is incrementally developed from prototypes. We allow both forms of prototyping: either enabling the user to create simple interactions and try out different scenarios (that can be thrown away); or alternatively providing code that is automatically generated, which can be then incorporated into other tools and developed further.

This approach provides much utility: users are able to create different types and styles of depictions that would not have been possible by the static tactile-graphic methods; highly complex interactions can be created that can haptically guide users through the depiction, so that they can be directed to interesting 'features', or their motion smoothed should users have involuntary muscle activity; the system can be used as part of a distance e-learning strategy, where new models can be created remotely and used locally; and finally, the users can be monitored and evaluated to understand their use which aids the improvement of the models.

While our focus in this article is to develop haptic representation of data for blind or partially sighted users through a rapid prototyping model, the system can be used to investigate new haptic interactions, or could be used to represent data haptically alongside other non-haptic visualizations.

This article describes our HITPROTO haptic data visualization tool (HDV) and it extends work presented in the haptic symposium conference [2]. This paper provides more information including details of HITPROTO implementation, examples of use, related literature and our evaluation process. We also provide reflection on the development and use of HITPROTO and present a summary of lessons learnt that can be used as a guide to researchers and developers of similar systems. Specifically, our contributions in this article are:

1. **Overview of current practices** for Haptic Data Visualization systems, and a discussion of related work for haptic prototyping and HDV (Sections 2 and 3).
2. The **design and implementation of the HITPROTO** haptic prototyping tool for HDV (Sections 4 and 5).
3. Three **different HDV examples** of varying complexity that were created with HITPROTO (Section 5).
4. Usability **evaluation of HITPROTO** that investigates whether support workers for blind or visually impaired users can utilise the system (Sections 6, 7 and 8). The participants are postgraduate students, which is justified by the observation that workers in the support units are often postgraduate students.
5. **Lessons learnt** for prototyping 3D Haptic Data Visualizations (Section 10).

2. The Haptic Data Visualization Process

We define Haptic Data Visualization (HDV) as the use of tactile devices or force-feedback technologies to convey information. The use of the tactile and force-feedback systems to present data to blind users is commonplace in the form of static tactile graphics, and is growing in dynamic haptic devices [3]. The designs that are created can be equivalent representations (similar in design from one modality to another) or can be new abstract designs that are specific to that domain [4]. The data is mapped into a haptic design that demonstrates relations within said data. The user can perceive these relations and understand and interpret the underlying information. A comprehensive review of designs for Haptic Data Visualization is presented by Panëels and Roberts [5].

The process of developing a sensory substituted view is similar for each of the senses and follows the dataflow paradigm of traditional visualizations [6]. Firstly, the developer decides what data is to be presented, selecting what is required through a *filtering* of the data. The data is then *mapped* onto haptic sensations to *display* each of the variables. These haptic variables determine how the user will perceive *value* and how the user will navigate the haptic representation.

HDV presents specific challenges to the engineer, because the haptic channel conveys less information simultaneously than the visual system, therefore developers need to simplify and abstract, or even idealise the information before mapping it to haptic variables. For example, a user may choose to demonstrate the overall trend of the data over time, in a haptic-graph representation, which maps *value* to *position*. But, there are many haptic variables that can be used to purvey quantity, such as the actuator position, the force-strength, the vibration frequency and the surface texture. HDV engineers also need to ascertain how the user will interact with this information. In a data visualization the user can modify any parameter to (say) alter the method of aggregation, or the scale of the plot. Likewise one of the advantages of using HDV rather than static tactile graphics is that the user can interactively change the display and alter the filtering, mapping or viewpoint of what is being represented.

In our work on haptic data visualization, we follow a two-stage approach, similar to the tactile graphics translation process, where the haptic data visualizations are prepared by a mentor or teacher, and then used by another person. We explain the tactile process in more detail in section 3.1.

3. Related Work

There are several related areas that impact on HDV that we review in the following subsections. These include tactile graphics and static representations; dynamic technologies; and finally interaction and haptic prototyping.

3.1. Tactile Graphics and Static Representations

In their seminal paper, Way and Barner [7] describe an automatic process of making tactile graphics from images – they concentrate on the use of image processing techniques to simplify the image before imprinting the tactile form. Several researchers have developed these ideas and created automatic systems (e.g., [8]). In addition, several institutes for the blind provide comprehensive books and reports that give guidance for the creation of appropriate tactile graphics (e.g., [9]). These resources provide practical advice, for instance Sheppard and Aldrich [10] suggest (1) eliminating the non-essential graphical elements, (2) substituting essential graphics (e.g., a haptic graphic of a spring may not be understood, but the physical object would be instantly comprehended) and (3) redesigning and simplifying the graphic. These guidelines also extend to the physical nature of the image, such as keeping lines greater than 2mm apart, or avoiding line labels.

Various technologies can be used to realise tactile graphics. These include: microcapsule paper (see Figure 1, left), thermo-form and vacuum-form or embossing. In addition, other tactile and tangible materials are used in classrooms, such as pins and rubber bands or Wikki Stix to create their own tactile diagrams (see Figure 1, right). The advantages of these low-tech solutions is that students can create their own charts and then get other people to touch them; it also enables them to understand and perceive concepts more effectively.

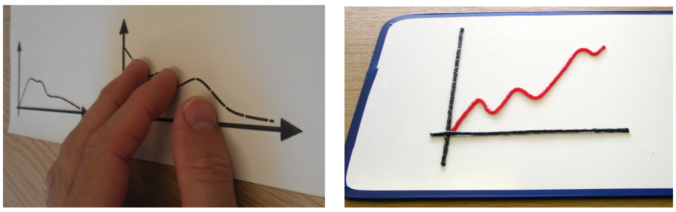


Figure 1: Left, photograph of swell paper, that swells up when heat is applied. Right shows wikkistix, which are strings doped with wax to make the string keep its form.

3.2. Tactile and Force-Feedback Technologies

Tactile and force-feedback technologies have been in development over the last 50 years. Nonetheless, it was not until the 1990s that the technologies became widespread. The growth of vibrotactile actuators in remote controls of home-games market and their inclusion in mobile phones and smartphones, in recent years, has led to their ubiquity. Many devices now achieve realistic *tactile* feedback. There are several papers that comprehensively review tactile technologies, such as [11].

On the other hand, *force feedback* technologies can also be used to represent data. These devices emerged to perform teleoperation in nuclear or subsea fields. One of the earlier applications in virtual reality was the GROPE project [12] for docking molecules. In fact, it is a good example of haptic visualization, where users could view molecules and investigate the forces of different molecule configurations. These devices developed into the force-feedback tools that are located in many haptic laboratories. For instance the PHANToM Premium or Desktop are capable of high resolution, while devices such as HapticMaster provide large workspaces and forces. More recently lower cost haptic devices have been developed, such as the PHANToM Omni and the Novint Falcon.

3.3. Prototyping Interactive Haptic Systems

The technique of 'prototyping' has been used in several fields of software development, but less so in the haptic domain, and even more rarely for HDV.

In the field of Virtual Reality (VR), several software tools exist for the creation of 3D models, but it is difficult to create interactive systems with them, and they have even less facility for developing haptic interactions. In fact, several researchers have investigated new languages to investigate 3D interaction techniques. Most do not deal with haptic properties, except for NiMMiT [13], a high-level notation system for multimodal interaction techniques. However, our focus is on interactions for haptic data visualization, rather than interactions for virtual reality.

More generally, there are several toolkits that support the *engineering of interactive systems*, especially over different (multimodal) devices, such as the iStuff toolkit [14], the Input Configurator (ICon) toolkit [15] or the OpenInterface (OI) framework [16]. Most of the prototyping tools mentioned above enable various multi-touch surfaces, mobile devices and other devices to be connected together. However, either they involve some programming, thus they are not accessible to non-experts, or it is unclear how haptics can be easily integrated into the systems.

Recently some researchers have designed tools for the prototyping of haptic or telehaptic applications. Rossi et al. [17] designed a tool for the rapid prototyping of haptic worlds built on the Matlab/Simulink platform where users can create a block diagram to create a VRML haptic world. Protohaptic [18] enables non-programmers to construct three-dimensional haptic models and the HAML-based Authoring Tool (HAMLAT) [19] extends Blender to allow non-programmers to create visual-haptic worlds. Kurmos et al. [20] uses the Scene Authoring Interface (SAI) to integrate haptics into an X3D authored virtual world.

However, all these prototyping environments focus on developing a haptic model of an environment and do not address the behaviour or interactions in this environment. On the other hand, the HAML framework [21], based on XML, does aim to provide a fast prototyping environment that hides the complexity of haptic programming. Another recent development is the HapticTouch toolkit [22] that provides a simple programming interface for the development of haptic tools. The goals of HapticTouch toolkit are similar to ours: to provide a simple interface. However, our work differs by focussing on haptic data visualization and on creating an environment for someone with no programming experience.

4. HITPROTO Toolkit

The purpose of HITPROTO is to assist developers in rapid prototyping of haptic interactions, with an emphasis on data visualization. As highlighted in the related work (Section 3), there are not many prototyping tools available for developing and testing haptic interactions. The few that do integrate haptics in their framework often describe the blocks using an input/output flow, which can be unintuitive when programming complex interactions. In contrast, HITPROTO attempts to ameliorate the technical complexities and provide an interface that is closer to a natural language (e.g., "*Wait for a button press, then add and start guidance*"). We hypothesise that in doing so, prototyping haptic interactions will become accessible to people with little or no programming knowledge. We also believe that by following this approach haptic interactions can be created faster by developers and designers, compared to learning the language and API required to program a device.

HITPROTO uses H3DAPI (h3dapi.org), an open-source haptics software development platform that uses OpenGL and X3D, and provides support for several haptic devices. It has been implemented in C++ with WxWidgets. H3DAPI allows users to build applications for different haptic devices and combines X3D, C++ and Python, offering three ways to program haptic applications. In theory, HITPROTO could be used with any devices supported by H3DAPI; however the current system has only been tested with the PHANToM Desktop, whereas support for more devices is planned for the future.

4.1. Block Diagram Design

We use a modular approach in HITPROTO, where users drag-and-drop components onto a canvas and connect them together to provide the logic for the haptic visualization. Parameters of the blocks can be set to describe specific behaviours. The arrangement of the blocks describes the semantic structure of the haptic interaction. This methodology was chosen to create an environment that non-technical users can operate. Our visual programming style draws inspiration from other visual programming environments, in particular the Lego Mindstorms NXT-G [23] software environment. For instance, in NXT-G users can create a diagram of three blocks that makes a robot *move forward*, *wait for 2 seconds* and finally *move in reverse*.

We utilize a three-step process to create the final haptic interaction (as shown in Figure 2). First, the user selects blocks to place on the canvas and links them together. They then adapt default parameters of the blocks to describe specific behaviours. Second, the block diagram is saved in an XML file (with a `.hit` extension). Third, these XML files are parsed using H3DAPI into the corresponding H3D Python code.

The Python code is then executed using H3DAPI to create the haptic interaction with the haptic device (such as the PHANToM Desktop). This means that the visual blocks in the canvas of HITPROTO are implemented independently of H3DAPI. This abstraction enables different parts of the system to be re-written in the future, e.g., a different graphical user interface could be built. Furthermore the use of the intermediary (`.hit` files) and the Python code affords other benefits. The `.hit` files can be saved and edited externally of HITPROTO and then re-loaded, as well as be easily shared between users. The use of Python files, rather than directly instantiating the different nodes in C++, allows the Python files to be utilized as skeleton code and incorporated in a bigger haptic system, or the Python files can be extended and adapted by a developer, separately from HITPROTO.

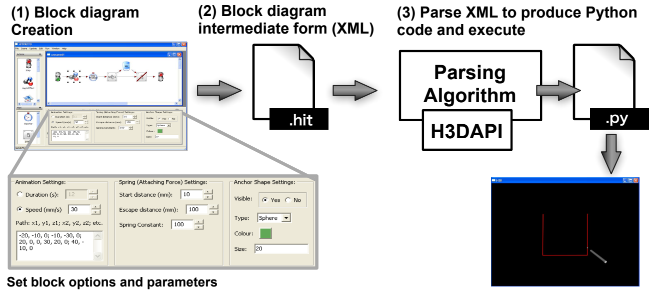


Figure 2: There are three main parts to the process: (1) The user makes the HDV scenario by connecting modular blocks together on the HITPROTO canvas (Section 4.3), (2) this information is saved in the `.hit` XML file (Section 4.4) and (3) HITPROTO generates a H3D Python file, and H3DAPI is used to execute the haptic world (Section 4.5).

4.2. Interface

The HITPROTO graphical interface contains four regions (see Figure 3): the menu bar; the left panel that contains the blocks; the middle canvas panel where the haptic program appears; and the bottom panel where users change the parameters of the blocks. The menu bar allows users to load and save files, open and include a scene object, run and implement the current selected scenario. The left panel contains the available blocks, which are divided into two lists. The upper list contains *Action* blocks and the lower list *Flow* blocks. Each block has a unique icon. The design of the icons were chosen so that they were relevant to the block's name and function. For instance, the *Switch* block is represented by a physical switch; the *Guidance_Add* block is pictured by a map.

The middle panel holds the diagram drawing canvas. The user can drag-and-drop blocks onto it. *Start* and *Stop* blocks are mandatory for all scenarios. The remaining blocks have a set of parameters which the user can edit to suit their needs. For example, 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, upon block selection. Executing the interaction diagram requires the user to appropriately link the blocks together from *Start* to *Stop* and then run the diagram.

4.3. HITPROTO Blocks

The logic of the final haptic program is determined by connecting the blocks on the canvas. Conversely, the implicit logic in the blocks determine possible block combinations. For instance, the switch block has one input and two outputs, which determines

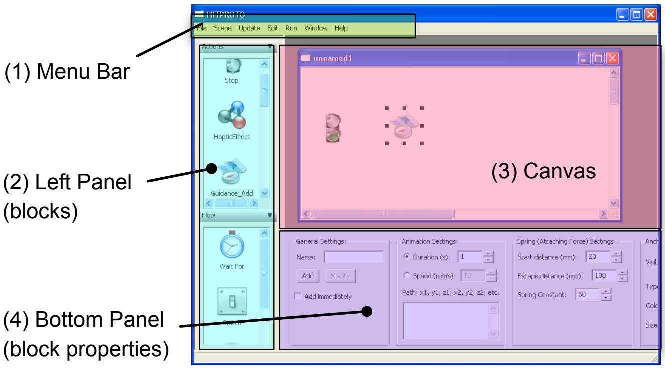


Figure 3: The HITPROTO interface consists of four principle regions: (1) the Menu bar, (2) Left Panel with blocks that can be dragged onto the Canvas, (3) the Canvas and (4) the Bottom panel where users can change parameter settings of the blocks.

how it is used in the canvas. This situation guards against some errors, but it is possible to create block diagrams that do not parse into runnable Python code. The error messages from the Python interpreter can be used to work out missing Blocks and/or parameters. In addition, users can unfortunately introduce semantic errors, e.g., by adding erroneous parameters in the blocks, which would be noticed during the final operation of the haptic interaction. We are currently developing HITPROTO to include more error checking.

Blocks fall into two categories: **Action blocks** that describe the addition, removal, creation and modification of haptic and guidance effects, and **Flow blocks** that control the flow of the data by listening to events. Table 1 includes a description of each block.

Blocks represent combinations of elements and functions that are available in H3DAPI. For instance, *Guidance_Add* and *Guidance_Control* combine several API elements, including a geometric representation (`H3D::Shape`), a spring force (`H3D::SpringEffect`), a time sensor (`H3D::TimeSensor`) and a position interpolator (`H3D::PositionInterpolator`) for the movement. The options and parameters for these effects are based on the same input parameters used to define them in the API and are set by the user, through the HITPROTO GUI. Other, simpler blocks, such as *Trash* or *Add_Modify* merely remove or add/modify specified objects. Likewise, the *Haptic Effect* block encompasses the different effects provided by the API, including a constant force (`H3D::HapticForceField`), magnetic lines (`H3D::MagneticSurface` and `H3D::LineSet`) or a spring effect (`H3D::SpringEffect`). As an example, a spring effect is defined by its position, the start distance of the effect, the escape distance and the spring constant.

Output values are directly integrated into the blocks. 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 operations, for example testing which keyboard key was pressed is performed with the *Switch* block. The *Switch* block effectively creates an implicit linking between the output of one block to the input of the other.

In addition, **Scene Objects** can be included in the world. These are X3D files that define a three-dimensional scene. Global 'containers' can store objects of those scenes, acting effectively as variables. Consequently, the 'containers' or variables 'selected', 'highlighted' and 'touched' (the last object touched by the device) are directly integrated within the suitable blocks (*Switch*, *Select*, *Highlight*, *Unhighlight* and *Trash*). As a result it is possible to create or remove haptic effects for specific scene objects.

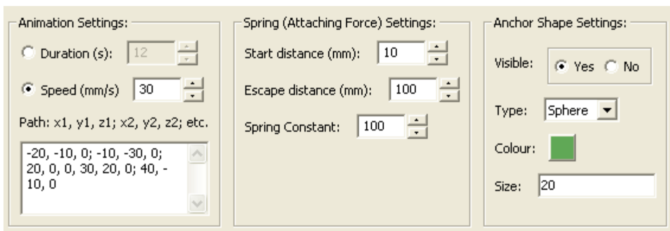
In practical terms, to generate a HITPROTO block diagram a user needs to decide which blocks are required, link them together and add appropriate parameters. It may be unclear to a new user which block should be used and what parameters to set. However, this would be true for a new user of any programming tool. We assist the user by a learn-by-example methodology and provide tutorial examples and descriptions. We used this approach in our experiments, by presenting a tutorial with several examples, each more complex than the previous. Finally, we provide a set of new tasks for the user to perform, based on what they have just learnt. This helps the user learn the three-step process (of creating the blocks, saving the `.hit` file and running the Python file) to create a working example.

4.4. Block Diagram Intermediate Form (XML)

The intermediary XML form (the `.hit` file) includes the order of blocks and the values for their parameters, as set by the user through the GUI. This XML code is therefore a direct encoding of the block-diagram and is created by iterating through the blocks and inserting the corresponding XML fragments into a string, before finally writing to file.

For example, the block diagram of Figure 4a, along with the associated parameters (Figure 4b), demonstrates a simple guidance interaction that starts as soon as the device gets attached to the guidance object, in this case a green sphere. The corresponding `.hit` file is shown in Figure 5.

(a) Block Diagram



(b) *Guidance_add* block parameters

Figure 4: Example of a guidance interaction diagram, that starts once the spring of the guidance object 'MR' gets active. The scenario is comprised of the *Start*,*Guidance_add*, *Wait_For*, *Guidance_Control* and *Stop* blocks.

Table 1: HITPROTO Blocks

(a) Action Blocks

Icon	*Description*
	**Stop** – compulsory block that delimits the end of the 'interaction scenario'.
	**Guidance_Add** – creates a guidance instance. Includes a spring to attach to the device and an anchor to visualize the spring and parameters such as path and speed/duration.
	**Guidance_Control** – enables the control of a guidance instance, by starting, pausing, resuming or stopping it.
	**Haptic Effect** – creates a chosen haptic effect. The available haptic effects are: SpringEffect, Magnetic Line(s) and PositionFunctionEffect (model-based). This is determined by a pull-down menu that changes the parameter set that the user can enter.
	**Add_Modify** – allows the addition or modification of an object. Previously removed object can be re-added.
	**Trash** – enables the removal of an object. Does not delete the object as it can be added back using Add_Modify.
	**Highlight** and **Unhighlight** – enables the haptic "highlighting" of an object by adding a spring to the object, making it magnetic or surrounding it with a magnetic bounding box. Removes the haptic "highlighting" of a named object (the name comes from the X3D Scene Graph). Multiple effects can be cleared in one step.
	**Select** – enables the tracking of the selected object by putting it into memory.

(b) Flow Blocks

Icon	*Description*
	**Wait For** – enables the interruption of a sequence of actions until a chosen event happens such as a haptic device/mouse button or a keyboard key being pressed/released, an elapsed time or the activation of a spring.
	**Everytime** and **Everytime_end** – enables the execution of a set of actions specified within the two blocks every time a chosen event occurs, such as haptic device/mouse button or keyboard key(s) pressed/released, elapsed time, haptic device touching an object and guided movement state.
	**Switch** and **Switch_end** – Checks if a condition is satisfied or not before executing a set of actions contained between the two blocks. Used after a Wait For or Everytime block. The Switch block has exactly two lines departing from it for each condition. Tests include *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* - used with the Everytime block for current position or elapsed time and *Comparison* - testing if specific values are equal.

The code demonstrates the sequential nature of turning the blocks into XML *elements*. Parameters set in the GUI are stored as the element's *attributes*. For example, the guidance path parameters, shown in the bottom left entry box in the parameters pane (of Figure 4b) can be seen as the attributes of element `<General>` of *Guidance_add* in line 4 of Figure 5.

```
1  <Start>
2    <line id="1">
3      <Guidance_Add position="120, 105"
           name="MR" addnow="1">
4        <General  path="-20, -10, 0; -10, -30,
             0; 20, 0, 0, 30, 20, 0; 40, -10, 0"
              speed="30"/>
5        <Spring k="100" startDist="10"
              escDist="100"/>
6        <Shape vis="Yes" type="Sphere"
              color="rgb(104, 170, 85)" size="20"
              />
7      </Guidance_Add>
8      <WaitFor position="195, 100" selection="4"
           condition="0" spring="MR"/>
9      <Guidance_Control position="270, 100"
          instance="MR">
10       <Start checked="0" />
11     </Guidance_Control>
12     <Stop position="345, 100"/>
13   </line>
14 </Start>
```

Figure 5: The .hit file of the interaction scenario shown in Figure 4. The variables and options set in the GUI are visible as each element's attributes.

4.5. Block Diagram Parsing and Python output

We parse the .hit file sequentially, and the parameters stored as XML attributes are passed to the respective Python code components.

However, code generation is not a linear process. This is because the labels and values may be defined at a lower position in the .hit file, and also there are dependencies in the Python code. In particular, the flow blocks *Wait For* and *Everytime* correspond to 'H3D-Python' classes that are listening to events, which in turn need to be initiated from the 'main' body of the Python file or indeed other classes. For instance, consider a simple sequence that adds a guidance object, and then waits for its spring to become active before the guidance object starts moving. The Python file would include the code to create an instance of the guidance object with the chosen parameters. This is located in the main body of the file. Within 'main', there are calls to a class that listens for the events of the guidance instance. In this latter class we include procedures to start the guidance when the spring becomes active.

The resulting Python file structure depends on the sequence of 'Action' and 'Flow' shapes. The 'Action' shapes code can either be located in the 'main body' or within a class listening to events, depending on whether a ‘Flow’ block precedes it. *Wait For* and *Everytime* shapes require their own class. *Switch* and *Switch_end* blocks are also a particular case. The algorithm checks conditional statements specified within the XML element and ensures that an appropriate sequence of ‘if-else’ is written. 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. Finally, 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.

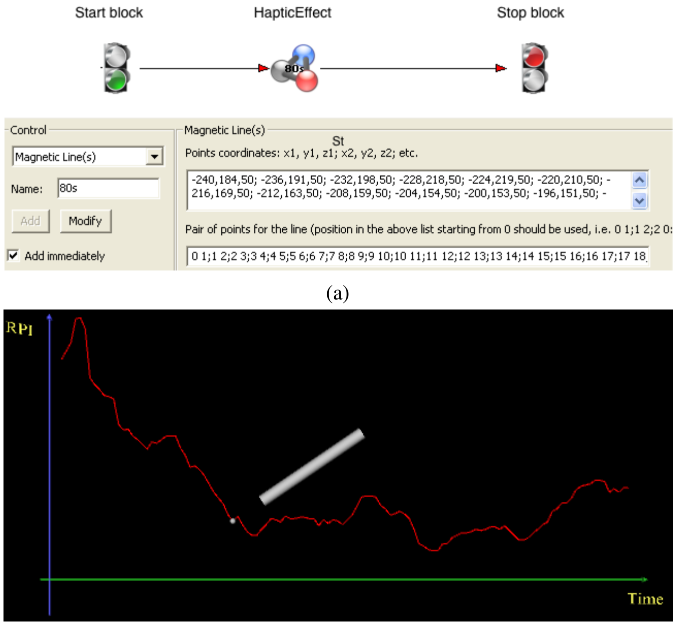
The Python code generated provides a runnable implementation and can be executed directly from HITPROTO’s interface, or the Python file run separately of HITPROTO. For the mentor/blind-user situation this may be enough, and it affords quick development and deployment. However the code can be reused and extended for the needs of another application. Because we are using standard H3DAPI Python code, it should be relatively easy for an experienced programmer with H3DAPI knowledge to integrate the generated code into another application.

5. HITPROTO Examples

Here we present a series of examples, demonstrating how HITPROTO can be used to create simple haptic interaction demonstrations, as well as more elaborate interactions for the exploration of scatterplots and line charts.

5.1. Haptic Line Chart using Magnetic Lines

The first scenario is an extension of the magnetic lines demonstration from the H3DAPI installation. Using the *Haptic Effect* block, which allows the creation of haptic effects including magnetic lines, a series of graphs were created based on real data. These demonstrate the retail price index (RPI) in the UK since 1980, see Figure 6. The data are input as a sequence of point coordinates. The axes are loaded as part of an X3D scene and, in this example, they do not have any haptic properties.



(a)

(b)

Figure 6: Example of creating a simple magnetic line chart using (normalized) real data, depicting the RPI in the UK for different decades. Figure 6a depicts the interaction diagram along with the options and parameters of the *Haptic Effect* block, while Figure 6b shows the line chart for the 2000’s. Data obtained from The Guardian, original source Office for National Statistics (ONS).

5.2. Haptic Line Chart using a Guided Tour Model

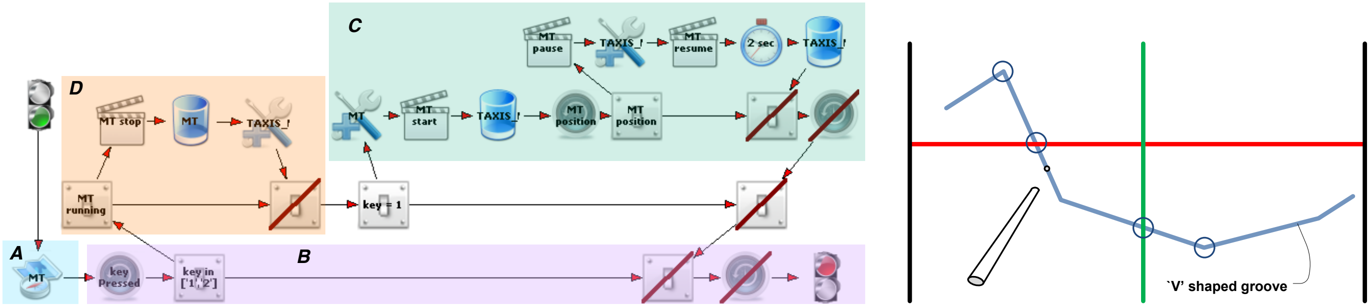
Line charts are one of the most common representations for statistical data. However, many challenges still remain for their exploration with non-visual techniques. We believe that guidance coupled with free exploration can contribute to building a better mental image of the chart. This scenario attempts to provide such an interaction, by employing the ‘museum tour’ metaphor [24, 25], where a user is driven along a predefined path and stops at predetermined points of interest, where they can roam freely to get a feeling of the surroundings before returning to the tour.

To create the guided haptic line chart we use the *Guidance_Add* block to attach the force over the path and control the speed of the guidance. We add a sphere to the tip of the haptic pointer to provide visual feedback for the location of the stylus. We create ‘points of interest’ at the maximum, minimum and inflection points on the graph. This is shown in Figure 7. The behaviour of the stylus on these points of interest is determined by the block named *Everytime*, which monitors the position of the haptic device. In addition we add a *Switch* block to test whether the point is passing over the point of interest, with a *Guidance_Control* block to specify what actions occur when this test is true. This enables the user to pause, roam around the point of interest and then resume the guidance. We remove the axis of the plots when the user is freely exploring locally (and the guidance is interrupted), and add them back into the world when the guidance is resumed.

5.3. Haptic Scatterplot using a Force Model

There are several ways to generate the forces for the haptic scatterplot and therefore there are potentially different ways to create an equivalent design. The challenge of displaying a scatterplot is very similar to haptic volume rendering. One technique could be to create a linear correspondence between the density of the visual points and the haptic transfer function and map this point density to the stiffness of the haptic device [26]. Alternatively, a proxy-based technique could be used [27]. We utilise a proxy based method for this example.

The process of analysing scatterplots consists of two tasks. First, understanding the overall trend of the data and to ascertain the size of the information, and second understanding specific features, such as outliers. Researchers sometimes call this process ‘eyeballing’ the graph, which is an important step in understanding the data [25]. The eyeballing process is somewhat



(a) Haptic Line Chart using a Guided Tour Model scenario diagram.

(b) Interaction Model

Figure 7: Line chart visualization using a Museum Tour metaphor with a V-shaped line and embossed axes on a chart surrounded by walls. The user is haptically guided along the engraved line and stopped for a given time, allowing to explore points of interest (the circles are located at the maximum, minimum and axis intersection points). In part A the guidance object is created using the *Guidance_add block*. In parts B, C and D the behaviour of the guidance interaction is specified. By pressing key '1' the museum tour mode is enabled, using *Guidance_control* (part C). Blocks *Everytime* and *Switch* constantly check whether the device pointer is on the specified points of interest. By pressing key '2' (part D) the tour mode is disabled, enabling free exploration.

different in haptic environment because the user has to actively explore the whole haptic world, however, the end goal is the same: to gain a quick understanding of the whole information, before drilling down into specific detail. This process is best described by Ben Shneiderman's mantra of 'Overview, zoom and filter, details on demand' [28]. Such details-on-demand could be represented to the user through the haptic modality itself (such as by representing value through vibrations or other haptic variables) or moved to another sensory modality, such as sound or spoken audible words.

In our haptic scatterplot example we focus on this overview task. In fact, it is good practice to separate different data visualization tasks into different haptic worlds [29, 30]. This is clearly explained in the handbooks that provide guides to produce effective tactile diagrams for blind users. For example, Eriksson and Strucel [31] write "When entering into more advanced mathematical problems it may become necessary to break up the picture into several parts and show them step by step in order to give a clear view of the problem".

We use Fisher's Iris dataset in our scatterplot example. This multivariate dataset describes the morphologic variation of three close species of Iris flowers; it was retrieved from the XmdvTool repository [32]. We generated both two-dimensional and three-dimensional charts to highlight the correlation of the flowers sepal length and petal length/width (see Figure 8b). Each dataset in the scenario is associated with a key on the keyboard, respectively keys 1, 2, 3. When the user presses a key, the haptic effect is added to the corresponding dataset (see Figure 8a). The user can understand a holistic perception over the location of the datasets, relative to each other, as well as their respective size by 'feeling' them successively or as a whole. This approach provides simplified and different views of the data [29].

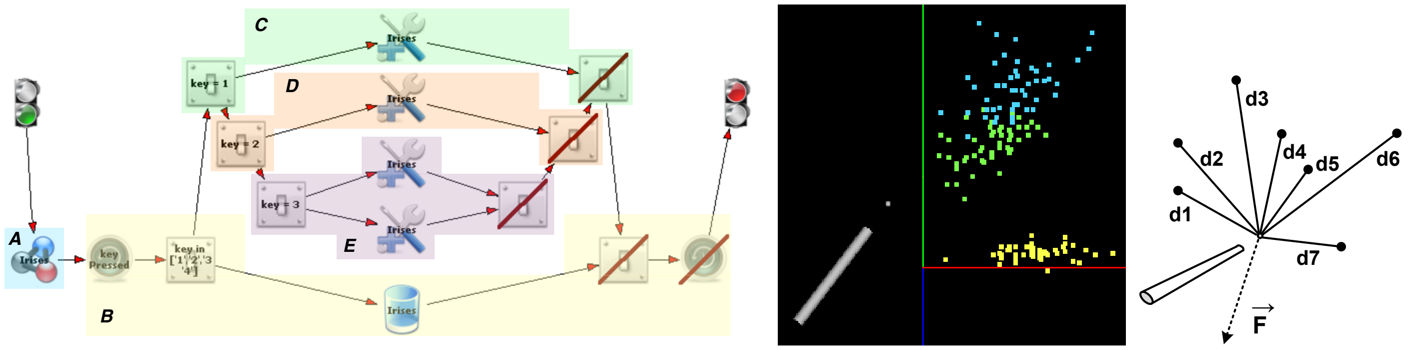
The *Haptic Effect* block defines the haptic effects. In particular, the haptic model for the scatterplot is defined in the 'Position Function Effect' field group of that block. The user can either specify the 3D components for the proxy object or apply a predefined force model, which can be applied to grouping nodes in the scene graph, according to the position of the haptic device. For this scatterplot scenario, we use a predefined model. We compute the resultant repulsive force 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 shown diagrammatically in Figure 8c. 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 $_i$.

$$\vec{F} = -k \times \sum_{i=1}^{n} \frac{1}{d_i} \times \vec{u_i} \quad . \qquad (1)$$

The use of this proxy model provides an overview of the point cloud and indicates the location of the other datasets when all the points of the dataset are used [25]. The closer the haptic stylus gets to a dense area of the scatterplot so the greater the force is applied on the stylus, whilst when the user is further away, the force on the stylus is less.

6. Usability Evaluation of HITPROTO

The main purposes of our evaluation was: first, to assess whether a support worker for blind or visually impaired users, could utilise HITPROTO to prototype simple haptic interactions; and second, to gain feedback over possible improvements to the tool. We chose to follow a formative evaluation approach. This was selected as the most appropriate method of assessment, and more precisely, an 'assessment test' as defined by Rubin [33]. The 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." (Chapter 2, p38). The PHANToM Desktop haptic device was used through the evaluation and both qualitative and quantitative measures were collected.



(a) Haptic scatterplot using a Force Model scenario diagram.

(b) View of the 3D display of the Iris datasetsl

(c) Diagram of the Force Model.

Figure 8: Scatterplot visualization using a Force Model. The HITPROTO scenario is shown in (8a). The haptic effect is first created (part A), then when a key is pressed this effect is either set for a particular grouping node (parts C, D and E) or removed (part B), depending on the key pressed. (8b) shows the 3D visual display of the enhanced Iris datasets used for the haptic visualization. (8c) shows a model of the forces used in the scatterplot visualization scenario. The resultant force is given in Equation 1 with d_i and $\vec{u_i}$ computed for the seven points.

Before the evaluation took place, we carried out a pilot study with two participants, in order to refine the assessment methodology, tutorial and tasks. The tasks' complexity was subsequently adjusted (the final tasks used for the evaluation are described in section 6.4). The supporting materials, in particular the tutorial and manual, were amended to reduce the use of technical terminology and make wording suitable for non-experienced users. In addition, we increased the length of the tutorial session. More detail on the pilot study, supporting material and resulting observations can be obtained from [2, 34].

6.1. *Evaluation Methodology*

During our formative evaluation sessions we followed a protocol of welcoming the users, introducing the project and describing the overall assessment procedure. We obtained their written consent for their participation and completed a background questionnaire with each participant. We described the role of the test facilitator, the role of the participant, the type of information to be gathered — making it clear it was not an assessment of their performance but of the tool — the training procedure, task overview, questionnaire description and purpose and estimated duration. The background questionnaire gathered information about their experience with visual programming tools and haptics, then users familiarised themselves with the PHANToM desktop by using the demonstrations distributed with H3DAPI.

The participants then underwent a training phase, which consisted of a step-by-step tutorial guided by the test facilitator. The tutorial included various interaction tasks to walk the participants through prototyping: how to create a new interaction diagram, manipulate the blocks and edit their parameters, connect blocks to create the interaction scenario, compile and execute the interaction diagram and test whether it achieves the given interaction scenario goals. At the same time, the participants were introduced to the blocks that they would use in the evaluation phase. At the end, a "check yourself" example was given so that the participants could try to create a diagram on their own, assisted if needed by the test facilitator, with the solution provided at the end.

Once the training was over, the participants were asked to complete a set of four tasks (these tasks are described in Section 6.4). They were encouraged to work without guidance, unless they did not understand the interaction description or were unclear of how to progress. During the assessment, the task completion time, task-goal success rate and whether (and how much) help was needed were recorded. Finally, a questionnaire was used, at the end of the tasks, to gather qualitative measures, collect participants' comments and record their experience with the tool, followed by a debriefing session.

6.2. *Task Analysis*

Lindgaard and Chattratichart [35] demonstrated that the number of participants in usability testing does not significantly correlate with the number of problems discovered, but that the task coverage does. Therefore, ensuring that the tasks cover the complete functionality spectrum of the toolkit was of high importance, compared to a large pool of participants.

In that respect, the evaluation tasks were designed around prototyping scenarios, involving subtasks such as choosing the correct blocks for the required interaction, drag-and-dropping them onto the canvas, connecting them appropriately, setting their parameters to obtain the intended behaviour and executing them to test whether the task is completed or not. For all the tasks, the successful completion criteria were whether the behaviour described in the task scenario has been achieved by the interaction diagram 'programmed' by the participant. The tasks were designed to have a gradually increasing difficulty and involved all the functional blocks available in the toolkit.

6.3. *Participants*

Due to the difficulty of recruiting designers, teachers or support workers for visually impaired students, at the time of development of the toolkit, postgraduates from the University of Kent were recruited instead. Our assumption was that the toolkit should be accessible to any person with no programming knowledge, no matter their background. These postgraduates fitted the profile of potential support workers for blind users. We used a 'convenience sample' of nine participants, as per Rubin's [33] recommendation, with a profile as shown in Table 2. The participants were postgraduate students and consisted of three males and six females. Their ages ranged between 22 and 29 years old and they had backgrounds in anthropology, archaeology, psychology, microbiology and actuarial science.

Table 2: The profile of the participants.

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 or beginner
Haptic Experience	None to some (e.g., interaction with products with limited haptic effects, such as in game controllers or vibrotactile devices in smart phones)
Visual programming experience	None or limited

6.4. Evaluation Tasks

The first task involved the creation of a magnetic square outline. The square should appear only after the button of the PHANToM is pressed.

In the second task the haptic stylus is led along a given path by an anchor object. The interaction should only start when the keyboard key 's' is pressed and should start at the current position of the PHANToM.

In the third task the haptic stylus is required to be guided along a given path by an anchor object. However, in this case the interaction should only start after the device is attached to the anchor object and the keyboard key 's' is pressed. If the PHANToM stylus gets detached from the anchor point or a different key (apart from 's') is pressed, then the scenario should end.

The fourth task reproduced the aforementioned 'Museum Tour' metaphor, where a visitor is guided along a path and stops at predefined points of interest, for a given time, before moving to the next item. The task was to generate a guided navigation, which commences once the device is attached to the anchor object. Once movement has started, each time the PHANToM passes from a set point of interest the interaction is paused for three seconds and then resumes. During those three seconds the PHANToM is allowed to move in a wider range.

7. Evaluation Results

We describe the time for task completion, success rates for completing the task and finally report the results of the questionnaire.

7.1. Time for task completion

Out of the nine participants, seven completed all the tasks, one did not complete the last task and one completed only the first task. The times were averaged over the number of participants who had completed each task, i.e., averaged over nine, eight, eight and seven participants, respectively, and the results are shown in Table 3.

Most participants completed the tasks within a relatively short period of time, with the average time increasing for each successive task. This overall trend is explained by the fact that the tasks increased in difficulty. Nonetheless, at an individual level (see Figure 9), this trend appears only for three participants with two participants even exhibiting the opposite behaviour. We attribute this behaviour down to the learnt familiarity of the tool, where the participants became more familiar and confident with HITPROTO as they progressed through the tasks, and hence became quicker at the tasks even with their increased difficulty.

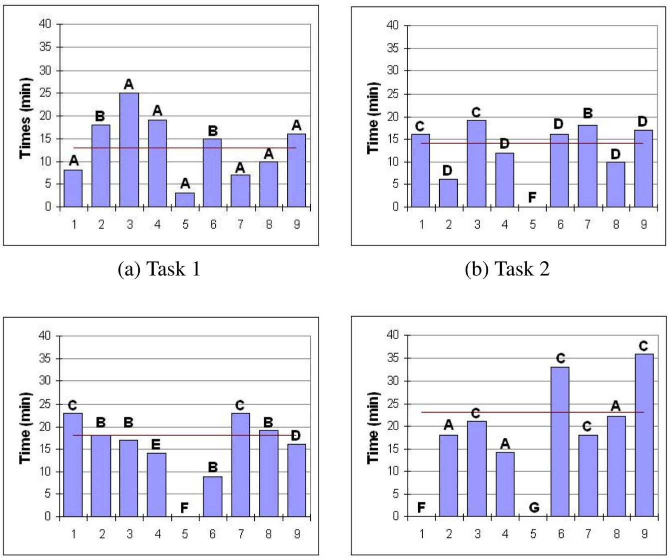
Table 3: Task Completion Time in minutes

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

7.2. Success rates for task completion

In the pilot study the first participant struggled with the terminology, understanding and remembering the block functionality and the methods of linking the blocks together. It may be that the training session was not long enough. Consequently, in the full evaluation we improved the tutorial material, increased the length of the tutorial session and allowed struggling participants to ask for help. The assistance that we provided 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 to the task was never directly explained. After the hints were provided the participants were left to work out the solution to the task.

Table 4 summarises the success rates for task completion, and depicts the participant count per task, for successful completion, taking into account the amount of help given. The categories include: success without help, success with minor help (e.g., page reference), success with major help (e.g., discussion including questions and explanations), minor errors without help (e.g., commencing the guidance at the device position when it was not required), minor errors with minor help, failure and task not attempted at all.



(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4

Figure 9: The charts show the ‘task completion’ time of each of the nine participants. The letters refer to the success rates for each participants. The meaning for the letters is included in Table 4, where the data is also summarised to give an overview of the success rates for each task.

The results indicate that earlier tasks were completed successfully with no or little help, whereas help was required for latter ones. This behaviour was expected because the latter tasks were designed to be more challenging than earlier ones. 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.

Table 4: Tasks success rates

Success reference	Description	Tasks 1	2	3	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

7.3. Results of the post experiment questionnaire

After the participants had completed the tasks they were asked to complete a questionnaire. The questionnaire consisted of two parts, one using the System Usability Scale (SUS) [36], which is used to evaluate the usability of the tool, and the second part that contained open-ended questions to collect more general feedback from participants. The average SUS score, rating overall usability, was 67%. Individual scores from participants ranged between 50% and 92.5%, except for one at 17.5%. It was a pity that this particular participant gave rather low scores to all the questions, and they did not seem to be interested in any computing software, neither did they want to spend time learning a new tool.

Table 5: Results from the open-ended questionnaire

Question	Positive	Negative	Other[a]
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

[a]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”

Table 5 summarises the main topics discussed in the open-ended questionnaire. Responses were grouped in three categories of *positive*, *negative* and *other* for indifferent or medium bias replies. Most of the participants found the toolkit’s functionalities useful and easy to use. Eight participants out of nine commented that the tool was easy to use, especially after some practice. Only one participant appeared indifferent to the toolkit, the same participant that gave a SUS score of 17.5%.

When participants were asked to list positive features of HITPROTO they focused on the simple interface, commenting favourably on the intuitiveness of the block diagram approach and the fact it allowed them to easily build interactions that were complex, or at least which seemed complicated in the first instance. When participants were asked to list negative aspects three reported none, whereas the rest made suggestions for improvements such as a grid to assist block placement, a zoom facility for finer adjustment on complicated diagrams and to improve the quality of the block icons. Two participants commented that some of the technical terminology that was used in the handbook could be simplified. Another suggestion was that the help facility could be improved, so replacing the need to refer to the tutorial. Also, enhanced tooltips for the block icons were suggested, along with a real-time inspection window and an error checking mechanism.

8. Discussion of Results

The usability evaluation showed that participants with no or very little programming skills understood the logic behind the dataflow language and could create basic haptic interactions. They managed this in a relatively short amount of time and with a limited training period despite their initial unfamiliarity with the tool. Overall, participants felt that the toolkit was easy to use and the interface was fairly intuitive. The blocks could be improved by using enhanced tooltips, error checking and an enhanced help functionality; features quite common in visual programming and system design software.

Overall, users anticipated that fabricating haptic interactions was a complex task. It is safe to assume that for users with no programming experience, having to learn to use H3DAPI with the supported programming languages would be problematic. It is reassuring that most participants completed even the most advanced tasks of our evaluation and that the comments in the post-experiment questionnaire were positive. This supports the conclusion that prototyping the required interactions was much easier than the participants first anticipated. These reinforce our belief 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. Therefore, haptic prototyping in this way not only opens up the potential for use of these technologies with a wider audience. In our particular area of interest, it should enable mentors to develop haptic data visualizations for blind users.

An important aspect that affected the ease by which participants completed the tasks was the comprehension and use of the tutorial. Arguably, with any system that employs visual programming for prototyping, practice and training are of paramount importance. This can speed-up or slow down the rate at which a user completes the task [37, 38]. Just as our pilot study demonstrated [2], the choice of examples, simplification of terminology and appropriate visual references to graphical elements (i.e., blocks) are necessary for the user's understanding of the systems operation. The current supporting material will be continually amended as we add functionality to our software.

9. Using HITPROTO with blind students

The formal evaluation in the previous section focused on the creation of the HDV interactions and the usability of HITPROTO in relation to potential support workers. Our aspiration is to continue to explore various HDV case studies produced with HITPROTO to gain more feedback from the community of visually impaired and blind users, as well as to modify the toolkit to offer any specific functionality derived from the feedback of these users. We are working with the help of the Bangor University Disabled Services of the Student Support Unit to use HITPROTO in practice and potentially to move the tool into their workflow. With this motivation in mind we have performed two further evaluations: first, with a user (who was not a participant in the previous studies) to create more HDV interactions; second, to explore how effective the new interactions are when provided to a blind user.

The new user, a postdoctoral computer science researcher, with little experience in haptics and data visualisation, used HITPROTO to create a series of HDV interactions. These included the designs as described in Sections 5 and 6 and a series of new interactions in the form of magnetic line-graph charts (see Figure 6). In addition, X3D objects, axes and annotations were added to the designs. Overall, the user commented favourably on the usability of the system and felt it was relatively easy to create new interactions.

With the blind participant, we performed a talk-aloud session (see Figure 10) of the guided tour (see Section 5) and the magnetic line-graphs, created by the aforementioned user. We asked the blind participant to describe what they were feeling, discuss other forms of representation they used or knew about and commented on the potential of HITPROTO. They were extremely positive over haptic visualization techniques and saw much potential in the tool, especially for e-learning and distance learning.

In this session, the blind participant made some interesting comments. First, they discussed how they explored information in general, and explained that they had used static examples such as thermoformed images. Second, they made a general comment saying that many tasks take them longer to perform in comparison to a sighted user. For example, they said that "it could take approximately four times longer to perceive any sensory substituted image", such as a thermoformed image, a version described in words, or even a Braille representation. This comment matched with our experience of their use: they took time to investigate the haptic environment and they moved over the haptic representations repeatedly to carefully understand the trends and positions of the values. Third, they saw the benefit of the tour example that led them to points of interest. Fourth, we discussed how annotation could be incorporated into the representations or these values could be represented by sounds or audio (as applied by Fritz [39] or Yu [38]). Although this is a preliminary investigation, this process has shown that it should be possible to include HITPROTO in the general workflow processes of blind users.

10. Lessons learnt

The development of systems such as HITPROTO is intricate and time consuming. The HITPROTO interface is a modular visual programming interface, where graphical blocks can be arranged on a canvas to provide the functionality of the system. The developer needs to have an understanding of many different

Figure 10: Photograph of a blind user, testing modified versions of the magnetic line chart example of Section 5.

aspects: they need to understand the haptic libraries and how the abstracted components of the block diagram can be arranged to predict and eliminate potential errors. Parsing the block diagram is not simple, and neither is it linear, where code generated from one module may depend on another module. This article, especially our evaluation section, demonstrates that we have successfully developed such a tool for haptic data visualization. However, there is still much to improve.

We have started to reflect on the development process. In this section we present a summary of lessons learnt. We reflect on the creation and use of the visual prototyping interface (and its modular design), and on the use of the Haptic Data Visualization process itself. This is intended to act as an initial guide to other developers. These 'lessons learnt' have been collated through discussions between us as researchers on this work, what we have learnt from reading other papers, and on reflection from our experiences in building HITPROTO, as well as from our evaluations.

10.1. HDV - Modular design

Points 1 to 7 inclusive describe aspects that prototyping tools for non-experts should include, whilst points 8 onwards would be useful in the design of Haptic Data Visualization prototyping tools.

1. **Visual dataflow programming**. We have found that the visual programming environment of HITPROTO enables novice users to develop complex haptic worlds that would not have been possible by them through programming [40]. We believe that visual programming methodologies will enable such haptic environments to become more readily accessible to a wider community.
2. **Medium granularity system**. Getting the right granularity is important: if it is too fine it becomes complex for novice users; if it is too large it is difficult for users to create expressive interactions. We chose a medium level of granularity of blocks in HITPROTO as at this level users are able to perform complex tasks whilst keeping the block diagrams at an abstraction that can be understood and manipulated by a non-programmer. This is in contrast to the fine grain systems of the interaction engineering tools or the newer multimodal collaboration environments (see section 3.3).
3. **Block Appearance**. To help with the understanding of the visual diagram, the icons for the blocks should be chosen to be self-explanatory (as much as is possible) [40]. Our design goal was to make the icons for the blocks understandable by mimicking real-world objects. This not only helps users to understand the available blocks but also helps with diagram readability.
4. **Alternative notation (block diagram and Python code)**. Accompanying visual programming tools with the generation of runnable code enables an expert to generate an initial prototype and then integrate and extend it with other systems. In HITPROTO the block diagrams generate the Python haptic model, which can be incorporated into other applications, or edited to customize the functionality.
5. Reuse of modules to **learn by example**. HITPROTO can save/open interaction diagrams (as XML `.hit` files), thus granting the possibility of establishing a library of interactions. This enables users to load similar designs and reuse the block diagrams, therefore providing 'reuse by example'.
6. Use **standard notations** for extensibility. HITPROTO uses both **X3D**, to load 3D worlds and **XML** for file management.
7. Use of **popular open source API** for better support and wider coverage. The use of H3DAPI enables HITPROTO to be extensible to other devices. In addition, there is an active community of developers and users of this library, which provides support and means bugs are fixed quickly.
8. **Default parameters** for better comprehension and faster development. HITPROTO currently does not include default parameter values in the blocks. Fully filled blocks with 'draft' values would improve the understanding and thus speed development.
9. **Usage 'patterns'**. Usage Patterns have been investigated for Information Visualization [41], it should be possible to develop similar patterns for HDV.
10. **Error checking**. Real-time error detection and feedback during diagram design or alternatively upon diagram compilation would speed up prototyping, increase the reliability of the produced interactions and assist users. Currently HITPROTO does not provide error-feedback to the users. Although we disallow connections of modules that would create logical errors, and we do some parsing checks for the block fields, it is still possible to accrue logical and syntax errors with HITPROTO's visual block editor. However, we are planning a more detailed error-checking parser.
11. **Block Extensibility** by enabling the addition of new action blocks (with the provision of the corresponding python

code or by encapsulating a diagram) and editing of currently available ones. Currently HITPROTO does not support every possible interaction, and a mechanism to create and add new blocks to the library would enable the prototyping tool to evolve and increase its interaction coverage.

10.2. Haptic Data Visualization

One of the main challenges for the support workers and mentors is to create effective haptic data visualization designs. But much like data visualization, there is no overarching theory of HDV and therefore no ideal design strategy to develop effective representations. As a result, it is not clear to a developer which mappings should be used to create the best interface. Perception and neurology research [42] has provided developers with indications regarding the limitations of each visual variable. In data visualization, researchers such as Bertin [43] provide some guidelines for data visualization, but there are few guidelines on designing effective haptic data visualizations. Indeed in the 2010 'theory of visualization workshop' we discussed the need for a multi-sensory theory of information visualization and called for researchers to "develop a theory of visualization, particularly focusing on the variables, that is extensible to other senses" [44].

At present each design needs to be evaluated separately. Design guidelines are gradually emerging [5, 45] and, it is hoped that, as the applied perception and cognitive science research grows, so the theories of HDV will grow from human factors and human computer interaction research [44]. In fact, many modalities can be used together. For example, it can be useful to represent haptic values along with audible signals. Research is also ongoing to evaluate how humans integrate signals between different senses, such as vision and haptics [46].

The situation with HDV is similar to sonification, where there is also no applied theory [47]. But, there is much that can be learnt from how one community displays the information in their modality, and often ideas can be transferred from one area to another. Although, rather than naïvely copying one design method it is better to create a representation of the data that is specific to the task that the user needs to carry out [4]. Hence drawing on Hermann's definition for sonification [48] and our experience with HITPROTO we suggest that HDV should have the following properties.

1. **Objective mapping**. There should be a clear mapping from the data to the haptic representation. I.e. relations in the input data should be mapped onto specific properties of the haptic device.
2. **Value**. The mapping should enable the user to understand the underlying data, and perceive the data quantity that the object and its sensation represents. Values may be represented through modalities (such as sound).
3. **Reproducible**. The representation is identical if the same data is loaded in another session or day.
4. **Different data** can be loaded to be represented by the haptic data visualization.
5. **Systematic interactions**. There is a logical set of interactions that allow the user to explore the three-dimensional worlds and understand the data. In addition, that the method of interaction is consistent across the same type of HDV representations. E.g., when a new dataset is loaded the information is manipulated in the same way.
6. **Exploration is encouraged**. The system should permit users to explore the representation. Indeed, from our experiments it was noticeable that blind participants need longer to understand the representations.
7. **Provide feature guidance**. It is difficult for a blind person to gain an overview of the information and to locate specific points of interest, therefore it is useful to provide guidance or touring mechanisms that lead the user to those points of interest (such as HITPROTO's *Guidance_add* and *Guidance_control* blocks).

11. Future work & Conclusions

There are many areas that can be improved in HITPROTO. One aspect is the block diagram editor. It would be useful to develop a more comprehensive error-checking parser, to allow the creation of new blocks that permit different visualizations and also to allow users to build their own blocks. Our evaluation and user experience testing also demonstrated that users wish to have better support when creating HDV. It would be possible to add 'hinting' mechanisms to the dataflow block diagram editor, or usage 'Patterns' for users to follow typical patterns.

One challenge, that broadly applies to all information visualization tools, is that it is difficult to input data. The challenge largely concerns the simplification of data and the modification of the data format to one that can be incorporated into HITPROTO. Currently data is included in a parameter field of a block, but we provide little support for data-simplification. Such support would aid the mentor to create different visualizations. Indeed, it would be possible to develop a module block that loads JSON files or remotely loads data.

One of the uses for HITPROTO is the teaching of mathematical concepts. Therefore, it would be good to provide other pedagogical aspects of learning, including allowing the evaluation of how users are improving in their understanding of different concepts. In this domain, HITPROTO could be also extended to enable e-learning or distance learning.

In summary, we have developed HITPROTO. The goal of HITPROTO is to allow users to rapidly prototype haptic interactions. We have used HITPROTO to create HDVs for blind users. We have performed an evaluation of the system to evaluate the usability by mentors, and present a preliminary study of the user experience of blind users. We have provided three block-diagram examples that demonstrate some of the capability of HITPROTO for HDV, and presented a set of lessons for aiding users to develop three-dimensional HDV prototyping tools, and HDV tools. It is clear that HDV has much potential, indeed the experience of our blind users and the enthusiasm of our participants supports the notion that these technologies have much potential, certainly as their use in the public domain continues to increase. The presentation of our lessons learnt provides guidance for other developers, and in the future may be incorporated into a more comprehensive set of guidelines and so help to develop a general theory of Haptic Data Visualization.

References

[1] El Saddik A. The potential of haptics technologies. Instrumentation Measurement Magazine, IEEE 2007;10(1):10–7.

[2] Panëels S, Roberts JC, Rodgers PJ. HITPROTO: a tool for the rapid prototyping of haptic interactions for haptic data visualization. In: Haptics Symposium. IEEE; 2010, p. 261–8.

[3] Roberts JC, Panëels S. Where are we with Haptic Visualization? In: WorldHaptics (WHC). IEEE; 2007, p. 316–23.

[4] Roberts JC. Visualization equivalence for multisensory perception: learning from the visual. Computing in Science Engineering 2004;6(3):61–5.

[5] Panëels S, Roberts JC. Review of designs for haptic data visualization. Haptics, IEEE Transactions on 2010;3(2):119–37.

[6] Haber RB, McNabb DA. Visualization idioms: a conceptual model for scientific visualization systems. In: Visualization in Scientific Computing. IEEE; 1990, p. 74–93.

[7] Way T, Barner K. Automatic visual to tactile translation, Part I: Human factors, access methods and image manipulation. In: Trans. Rehab. Eng.; vol. 5. IEEE; 1997, p. 81–94.

[8] Wang Z, Li N, Li B. Fast and independent access to map directions for people who are blind. Interacting with Computers 2012;24(2):91–106.

[9] Miller I, Pather A, Muilbury J, Hasty L, O'Day A, Spence D. Guidelines and standards for tactile graphics, 2010. Joint BANA/CBA Tactile Graphics Committee, brailleauthority.org/tg/; The Braille Authority of North America; 2010.

[10] Sheppard L, Aldrich FK. Tactile graphics: A beginner's guide to graphics for visually impaired children. Primary Science Review 2000;65:29–30.

[11] Benali-Khoudja M, Hafez M, Alexandre JM, Kheddar A. Tactile interfaces: a state-of-the-art survey. In: ISR'04. Paris; 2004, p. 721–6.

[12] Brooks Jr FP, Ouh-Young M, Batter JJ, Kilpatrick PJ. Project grope - haptic displays for scientific visualization. In: ACM SIGGRAPH; vol. 24. ACM Press; 1990, p. 177–85.

[13] De Boeck J, Vanacken D, Raymaekers C, Coninx K. High-Level Modeling of Multimodal Interaction Techniques Using NiMMiT. Journal of Virtual Reality and Broadcasting 2007;4(2).

[14] Ballagas R, Ringel M, Stone M, Borchers J. iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments. In: Computer Human-Interaction (CHI). ACM Press; 2003, p. 537–44.

[15] Dragicevic P, Fekete JD. Support for Input Adaptability in the ICON Toolkit. In: Multimodal interfaces (ICMI). ACM Press; 2004, p. 212–9.

[16] Lawson JYL, Al-Akkad AA, Vanderdonckt J, Macq B. An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. In: Symposium on Engineering Interactive Computing Systems (EICS). ACM Press; 2009, p. 245–54.

[17] Rossi M, Tuer K, Wang D. A new design paradigm for the rapid development of haptic and telehaptic applications. In: Conference on Control Applications. IEEE; 2005, p. 1246–50.

[18] Forrest N, Wall S. Protohaptic: Facilitating rapid interactive prototyping of haptic environments. In: Workshop on Haptic and Audio Interaction Design (HAID). 2006, p. 21–5.

[19] Eid M, Andrews S, Alamri A, El Saddik A. HAMLAT: A HAML-Based Authoring Tool for Haptic Application Development. In: EuroHaptics. Springer; 2008, p. 857–66.

[20] Kurmos L, John NW, Roberts JC. Integration of Haptics with Web3D using the SAI. In: Conference on 3D Web Technology (Web3D). ACM Press; 2009, p. 25–32.

[21] Eid M, Alamri A, El Saddik A. MPEG-7 Description of Haptic Applications Using HAML. In: IHaptic Audio Visual Environments and their Applications (HAVE). IEEE; 2006, p. 134–9.

[22] Ledo D, Nacenta MA, Marquardt N, Boring S, Greenberg S. The HapticTouch toolkit: enabling exploration of haptic interactions. In: Conference on Tangible, Embedded and Embodied Interaction (TEI). ACM; 2012, p. 115–22.

[23] Kelly J. Lego Mindstorms NXT-G Programming Guide. Springer; 2010.

[24] Panëels S, Roberts JC. Haptic Guided Visualization of Line Graphs: Pilot Study. In: Workshop on Haptic and Audio Interaction Design (HAID), poster proceedings. 2007, p. 5–6.

[25] Panëels S, Roberts JC, Rodgers PJ. Haptic Interaction Techniques for Exploring Chart Data. In: Workshop on Haptic and Audio Interaction Design (HAID). Springer; 2009, p. 31–40.

[26] Avila RS, Sobierajski LM. A haptic interaction method for volume visualization. In: VIS '96. IEEE; 1996, p. 197–204.

[27] Lundin K, Gudmundsson B, Ynnerman A. General proxy-based haptics for volume visualization. In: Eurohaptics. 2005, p. 557–60.

[28] Schneiderman B, Plaisant C. Designing the user interface. Addison-Wesley Longman.; 1998.

[29] Roberts JC, Franklin K, Cullinane J. Virtual Haptic Exploratory Visualization of Line Graphs and Charts. In: The Engineering Reality of Virtual Reality (Electronic Imaging Conference). IS&T/SPIE; 2002, p. 401–10.

[30] McGookin DK, Kildal J, Brewster SA. New views on haptic graph visualisation. CHI Workshop In Hands on Haptics: Exploring Non-Visual Visualisation Using the Sense of Touch; 2005.

[31] Eriksson Y, Strucel M. A guide to the production of tactile graphics on swellpaper. The Swedish Library of Talking Books and Braille; 1995.

[32] Ward M. Xmdvtool: integrating multiple methods for visualizing multivariate data. In: Visualization. IEEE; 1994, p. 326–33.

[33] Rubin J. Handbook of Usability Testing: how to plan, design and conduct effective tests. Wiley Technical Communications Library; John Wiley and Sons, Inc.; 1st ed.; 1994.

[34] Panëels S. Development and prototyping of haptic interactions for data exploration. Ph.D. thesis; University of Kent; 2010.

[35] Lindgaard G, Chattratichart J. Usability Testing: What Have We Overlooked? In: Conference on Human Factors in Computing Systems (CHI). ACM Press; 2007, p. 1415–24.

[36] Brooke J. Usability Evaluation in Industry; chap. SUS - A quick and dirty usability scale. Taylor & Francis; 1996, p. 189–94.

[37] Halbert D. Programming by example. Ph.D. thesis; University of California, Berkeley; 1984.

[38] Yu W, Brewster S. Multimodal virtual reality versus printed medium in visualization for blind people. In: Conference on Assistive technologies (Assets). ACM; 2002, p. 57–64.

[39] Fritz J, Barner K. Design of a haptic data visualization system for people with visual impairments. IEEE Transactions on Rehabilitation Engineering 1999;7(3):372–84.

[40] Green T, Petre M. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. Journal of visual languages and computing 1996;7(2):131–74.

[41] Heer J, Agrawala M. Software design patterns for information visualization. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):853–60.

[42] Heller M, Ballesteros S. Touch and Blindness: Psychology and Neuroscience. Taylor & Francis; 2005.

[43] Bertin J. Semiology of Graphics. The University of Wisconsin Press; 1983.

[44] Roberts JC, Walker R. Using all our senses: the need for a unified theoretical approach to multi-sensory information visualization. VizWeek 2010 Workshop: The Role of Theory in Information Visualization; 2010.

[45] Reiner M. Seeing through touch: The role of haptic information in visualization. In: Gilbert JK, Reiner M, Nakhleh M, editors. Visualization: Theory and Practice in Science Education; vol. 3 of *Models and Modeling in Science Education*. Springer; 2008, p. 73–84.

[46] Takahashi C, Diedrichsen J, Watt S. Integration of vision and haptics during tool use. Journal of Vision 2009;9(6).

[47] Kramer G, Walker B, Bonebright T, Cook P, Flowers J, Miner N, et al. Sonification report: Status of the field and research agenda. Tech. Rep. 444; University of Nebraska - Lincoln, Faculty Publications, Department of Psychology; 2010.

[48] Hermann T. Taxonomy and Definitions for Sonification and Auditory Display. In: Conference on Auditory Display (ICAD). 2008, p. 1–8.