HITPROTO: a Tool for the Rapid Prototyping of Haptic Interactions for Haptic Data Visualization

Sabrina A. Panëels*
School of Computing
University of Kent, UK

Jonathan C. Roberts†
School of Computer Science
Bangor University, UK

Peter J. Rodgers‡
School of Computing
University of Kent, UK

ABSTRACT

The use of haptic devices is becoming widespread, particularly with their growth in the home-games market and on mobile phones. Likewise, the number of haptic devices and APIs is increasing. However, it is still difficult to program and develop haptic applications. Consequently, there is a need to hide the complexity of programming a haptic application or interactions by providing prototyping tools or frameworks. In particular, one growing application area is haptic data visualization (also called haptification): the use of haptic devices to represent and realize information and data. This paper presents a visual prototyping tool for haptic interactions for data visualization. Users can easily and quickly create haptic interactions through a visual programming interface with tunable parameters. This paper also presents a user evaluation aimed at demonstrating that non-programmers can create simple haptic interactions and which also indicates potential improvements to the tool.

Index Terms: H.5.2 [Information Interfaces and Presentation (I.7)]: User Interfaces (D.2.2, H.1.2, I.3.6)—Haptic I/O, Prototyping; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

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

1 INTRODUCTION

Haptics is a growing research area. It is also reaching the consumer market through the gaming industry. Consequently, the range of available haptic devices is increasing, from custom devices developed by research laboratories to commercial devices. For instance, high resolution devices such as the PHANTOM Omni are suitable for medical applications or research, whilst devices such as the Nintendo Wii and the Novint Falcon are aimed at the home market. However, 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.

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. 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.

*e-mail: sap28@kent.ac.uk
†e-mail:j.c.roberts@bangor.ac.uk
‡e-mail:p.j.rodgers@kent.ac.uk

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. For example, various applications have been developed that convey chart data or maps to visually impaired people; vector fields or flight information for pilots have been displayed haptically, which is useful as the visual display is already heavily loaded with information, see [26] for an overview.

However, the use of haptic interactions for visualization is not widespread [26, 29]. 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 paper presents a visual prototyping tool for haptic interactions (HITPROTO). 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 2 we discuss various related work for prototyping interactions and more specifically haptic interactions. In Section 3 we describe the design of the prototyping tool HITPROTO. Section 4 presents a usability evaluation of the tool. Finally, in Section 5 we conclude and give some directions for further work.

2 BACKGROUND & RELATED WORK

Our motivation is to create a tool that provides rapid prototyping of haptic interactions for haptic data visualization. In this section we discuss the two principal concepts: (1) rapid prototyping of applications and interactions, especially haptic interactions and (2) haptic data visualization.

2.1 Rapid Prototyping of Applications and Interactions

In software engineering, the term prototyping refers to the rapid development of a software solution. There are typically two forms: firstly throw-away prototyping where subsequent systems are created and then discarded in turn before the final development; and secondly evolutionary prototyping, where the user refines the prototype in an ongoing process before it turns into the final system. In our work, our focus is to develop a platform that allows developers to perform both of these forms of prototyping, the first through the rapid development of simple haptic interactions, and the second through the provision of automatically generated code that can be built-upon by developers.

2.1.1 General prototyping

In the field of Virtual Reality (VR), there are various tools that allow developers to easily create 3D models. Tools like Blender [2], 3ds Max [1] and Rhino [4] 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 the 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. [18] and the 'Interaction Framework For Innovation' (IFFI) by Ray and Bowman [28]. Recent work by DeBoeck et al. [12] has proposed a high-level graphical notation called NiMMiT to specify multimodal interaction techniques; thus including haptics. This software not only allows the design at a high-level but also automatic execution. However, our focus is on interactions for haptic data visualization rather than interactions for virtual reality.

More generally, Ballagas et al. [8] developed the iStuff toolkit, which "support[s] user interface prototyping in ubiquitous computing environments". However as the focus is on lightweight wireless devices, iStuff does not readily integrate haptic devices, such as force-feedback devices. Similar to Ballagas et al., Dragicevic and Fekete [14] implemented the Input Configurator (ICon) toolkit, which aims at dealing with 'Post-WIMP' interaction techniques by allowing high-level 'input adaptability'. However, it is unclear how complex interaction techniques, such as those involving haptics, would be easily integrated. Similarly, Huot et al. [20] developed the MaggLite toolkit for fast and interactive design of post-WIMP user interfaces which uses the ICon notation. However the toolkit is restricted to 2D presentations.

Navarre et al. [24] 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. [7] created the FlowStates toolkit, which combines ICon with the Java Swing extension SwingStates, to allow easy prototyping of complex interaction techniques. However, these techniques involve some programming, while our work aims at allowing rapid prototyping for non-programmers. Serrano et al. [32] highlights 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". Therefore they developed the OpenInterface (OI) framework, which components can be assembled to create a 'pipeline' for the definition of a multimodal interaction using the graphical environment OIDE [32]. As OIDE has several shortcomings, including inflexible design and not enough influence of non-developers, Lawson et al. [22] developed the SKEMMI graphical interface. Informal evaluations demonstrated successful use of the framework to prototype various applications. However, it is unclear how haptics can be easily integrated into this framework.

2.1.2 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 especially force-feedback devices would be easily supported. Rapid prototyping of haptic worlds has also been the focus of various researchers. Rossi et al. [30] 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 [19] 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 7 participants, including novices and experts in using the PHANTOM device, showed that the participants could construct 3D models within the time allotted. Kurmos et al. [21] 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 [15] 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 following user's requirements. After defining the structure and description schemes for the language, Eid et al. [16] developed the HAML-based Authoring Tool (HAMLAT) to allow non-programmers to create visual-haptic worlds, by extending Blender to support haptics. However, the preliminary work restricts itself to static scenes with no dynamic behaviour. Our tool on the other hand, deals with the interactions and thus the dynamic behaviour.

De Felice et al. [13] present an authoring tool to design the haptic/acoustic user interface of a VE explored by visually impaired people. The tool uses a model where the virtual scene is a series of scenarios, containing active objects (scene objects with acoustic and/or haptic properties) and guided paths. The visual editor allows the specification of object behaviour by assigning properties and events (translation/rotation). 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.

In the tactile domain, several tools have been developed to facilitate the prototyping of vibrotactile icons. MacLean and colleagues developed the Hapticon Editor [17] and more recently extended it into the Haptic icon Prototyper [33]. 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. [34] also developed a Haptic Icon Designer application for the design and playback of vibrotactile icons, for information display via floor surfaces. The application allows the design of icons in terms of short-time stimulus using frequency, duration, harmonic content, roughness, and amplitude temporal envelope and of longer-time structures using a musical phrase metaphor with rhythm, duration, note amplitude and repetition parameters. A pilot study conducted with 8 participants and 8 vibrotactile icons developed with the interface led to a correct identification rate of 55%. In particular, Lee et al. [23] based their prototyping tool on a musical metaphor to avoid the user dealing with the low-level specifications of the vibrotactile signals and thus making the tool accessible to a wider audience.

As for commercial tools, the Immersion Studio [3] 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 [10] and for both teleoperation research and haptic device development at Georgia Tech University [9]. LabVIEW has been used extensively in designing control and analysis solutions in the area of engineering technology and education. However, as it aims at engineers and scientists, the graphical no-

tation is quite low-level and focuses primarily on signal acquisition, processing and analysis rather than designing computer interactions, in particular 3D haptic virtual interactions.

2.2 Haptic Data Visualization

Haptic data visualization (or 'haptification') is the process of displaying data haptically; it gives users an understanding of information through a haptic interface [29]. The haptic visualization display can be considered to consist of three parts. First a model is created that contains the haptic information that is going to be perceived. The generation of the model closely follows the visualization process, where the data is enhanced (a subset of the data is selected, and processed such that it is suitable to be displayed) and mapped into an abstract/haptic model (the data is transformed into an appropriate haptic model). Thus it may be that the information may be simplified or averaged. Second, haptic rendering is used to compute the forces required to realize the model. Third, an electromechanical device is used to exert the forces that the user perceives. For an in-depth review of haptic data visualizations we refer the reader to "Review of Designs for Haptic Data Visualization" [26].

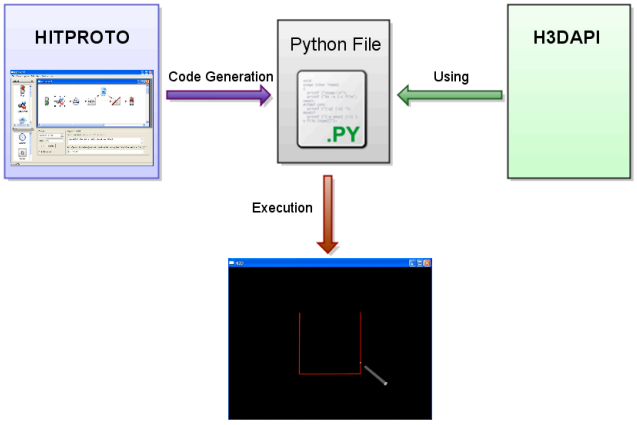


Figure 1: Diagram showing the different components involved in the creation and execution of an interaction scenario.

3 HITPROTO: Prototyping Haptic Interactions

HITPROTO allows developers to perform rapid prototyping of haptic interactions, with an emphasis on data visualization. As highlighted in the preceding related work section, 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 input/output flow which can be unintuitive to program complex interactions. In contrast, our tool 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"). We hypothesize 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.

3.1 Design

HITPROTO is based on the H3DAPI [5]. 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. The HITPROTO tool aims to significantly reduce the difficulty of designing haptic interactions.

The tool draws inspiration from various visual programming environments, and in particular the Lego Mindstorms NXT software based on LabVIEW. Mindstorms NXT is a visual programming environment that allows programming 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, creation and modification of haptic and guidance effects) and flow blocks (*Wait For*, *Everytime* and *Switch*) that allow controlling the flow of the data by listening to events (e.g. "Wait for the device's button to be pressed") or 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 generated in Python code which can be directly executed or that could further be used as a start code skeleton to extend the interactions or to learn the H3DAPI.

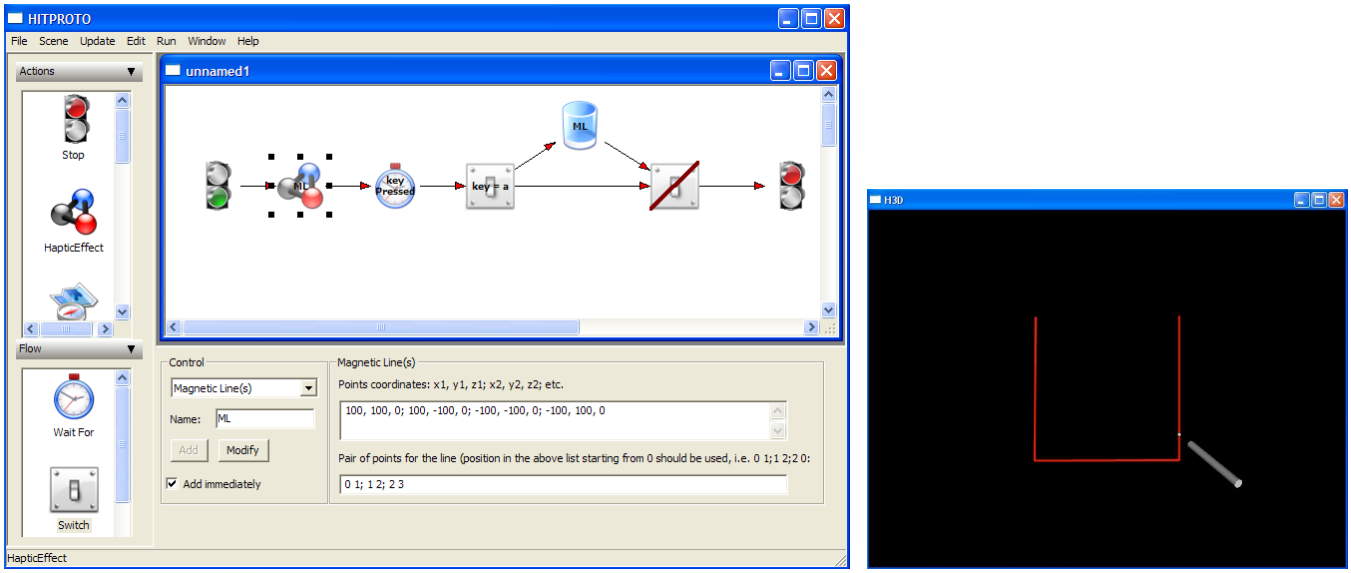
3.2 Implementation

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. The blocks and their parameters are based on elements and functions available in the API. After users link these blocks in an interaction diagram and start execution, each block is parsed and the corresponding Python code is generated, then executed with the H3DAPI, see Figure 1. Python provides an easy way to interface the H3DAPI. Therefore, we chose to generate Python code, 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 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.

3.3 Using HITPROTO

The tool is divided into four regions (Figure 2(a)): a menu bar (i.e. Save or Open File), a left panel, a canvas and a bottom panel. The left panel contains the block shapes, divided into two expandable list menus, namely 'Actions' and 'Flow' blocks. The 'Actions' blocks available so far include: *Guidance_Add* and *Guidance_Control* for guidance interactions, *HapticEffect* to add haptic effects, *Highlight* and *Unhighlight* to highlight an object haptically, *Select* to store an object into memory, *Add_Modify* to add or modify a created instance or a scene object and *Trash* to remove an object from the scene. The 'Flow' blocks contain *Wait For*, to wait for an occurrence of a specified event before proceeding with the rest of the actions, *Everytime* to repeat a sequence of actions each time a given event is received and *Switch* to test conditions on the received events. The block shapes are represented by icons, whose image reflects the block's 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.

[1] X3D [6] 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.



(a) Interaction scenario diagram.

(b) Corresponding 3D scene with H3DAPI.

Figure 2: These figures represent the scenario for creating three magnetic lines, which are removed if the first keyboard key pressed is 'a'. Figure 2(a) illustrates the interaction diagram that prototypes this behaviour, while Figure 2(b) shows the resultant 3D scene.

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 to attract the user). These parameters are displayed in the bottom panel, upon block selection. Testing the constructed interaction diagram requires appropriately linking the shapes from *Start* to *Stop* and then running the diagram. Also, for line chart visualization, a module allows the automatic creation, from a set of points, of an X3D scene, which the user can load and for instance, create interactions for it such as guidance (see Section 3.4).

3.4 Examples

As can be seen from the previous section, prototyping an interaction is achieved by connecting together the suitable blocks, setting the appropriate parameters and executing the diagram. Two examples are shown in Figures 2 and 3, which illustrate the creation of an interaction scenario.

The first scenario (Figure 2) was created as an extension of a demonstration from the API 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 2(a)) 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 remove the magnetic lines from the scene.

The second scenario (see Figure 3) was created in order to reproduce an interaction that has been presented using the prototyping language NiMMit [12]. In this interaction scenario, the device's collisions with 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 in the centre of the object or making the object's surface magnetic. In addition, every time the device's button is pressed, we select the current highlighted object if it is not already selected.

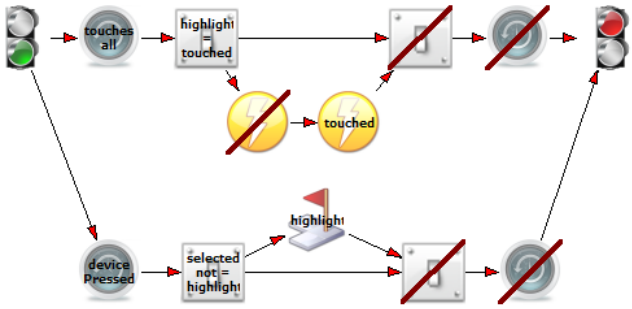


Figure 3: A highlight by touching and select by pressing the device button scenario, similar to DeBoeck et al. [12].

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 using a repulsive force based on point concentration to convey a 3D overview of scatter plots have been developed using the tool [27]. A simpler example of the 'museum tour'[25], which deals with the movement interaction only, is illustrated in Figure 4. 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 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 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 al-

low the user to roam around for a given time before resuming the guidance.

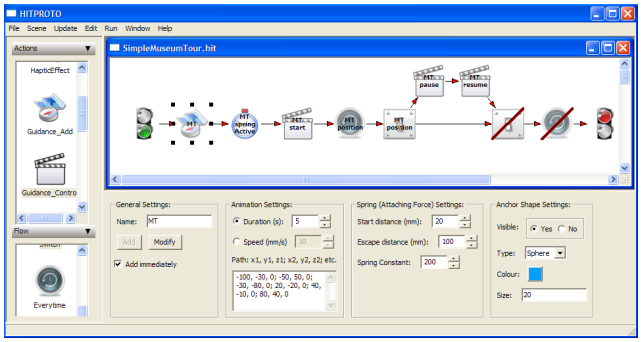


Figure 4: The museum tour interaction diagram: the user is led along a predefined path and at chosen points of interest, the tour is paused, to allow the user to explore the surroundings before resuming.

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 [13] to a more complete set of haptic interactions [11]. As a limited set of interactions is applicable to the haptic modality [11], a case-by-case approach has been judged feasible and adopted.

4 Usability Evaluation

The purpose of our evaluation was to assess whether a user can use the tool to prototype haptic interactions and what improvements can be made. Therefore, a formative testing was selected as the most appropriate type of testing, and more precisely, an 'assessment test' as defined by Rubin [31]. 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." (Chapter 2, p38).

The evaluation procedure is set-out below; in this evaluation, both qualitative and quantitative measures were collected and the PHANTOM force-feedback device was used throughout this evaluation. We first initiated a pilot study, which provided useful feedback as to our training and evaluation techniques, subsequently followed by a full study.

4.1 Procedure

Before performing the tasks, the participants were first asked to complete a background questionnaire to gather information about their experience with visual programming tools and haptics. A familiarization with the haptic device phase followed using a set of the API standard demonstrations. Then the participants underwent a training phase, which consisted of a step-by-step tutorial guided by the expert leading the test session. The tutorial included various interaction scenarios to walk the participants through their prototyping: how 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. At the same time, the participants were introduced 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 creating a diagram on their own, with the solution provided at the end as well as help on demand.

Once the training was over, the participants were asked to complete a set of four tasks. They were encouraged to work without guidance, unless they did not understand the interaction description or were unclear of how to progress. During the tasks several quantitative measures were taken: these included the time to perform the task as well as the success rate, including information on whether (and how much) help was needed to achieve the task. A questionnaire was used, at the end of the tasks, to gather qualitative measures, followed by an interview about the participants's comments and experience.

4.2 Pilot Study

The pilot study followed the main procedure described above, in Section 4.1. 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 3.4, 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 3.4. 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 participants were given four tasks (or interaction scenarios) to perform. Scenario 1 was to create a guidance interaction, which started both after the device was attached to the moving anchor and after a certain key on the keyboard was pressed. This scenario's difficulty was in understanding that it was possible to add two *Wait For* blocks after one another. Scenario 2 was to create a guidance interaction which started once the device was attached to the anchor object. The guidance had to be paused (if it was moving) or resumed (if it was already paused) each time the device's button was pressed. This scenario's difficulty lay in finding the right test with the *Switch* block, which had not been introduced in the tutorial. Scenario 3 was a simplified version of the 'Museum Tour', as explained in Section 3.4, where the guidance would pause and then resume at given points of interest. This scenario's difficulty involved getting the right combination of the *Everytime* and *Switch* blocks with the appropriate option, also not presented during the tutorial. Scenario 4 was a simplified version of the scatter plot overview interaction [27]. The first three task scenarios had 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 experiment. Both were postgraduate students with no experience with haptics, programming or visual programming tools and were paid for their participation.

4.2.1 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.

4.3 Changes after the 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 pilot's first and third task scenarios. These tasks, as explained in Section 4.2, 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).

4.4 Experiment

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. 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, with backgrounds including anthropology, archeology, psychology, microbiology or actuarial science.

4.4.1 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 1. 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 1 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 5), 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 three 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). We believe the high values for participant 2 were partly affected by the successive crashes encountered.

Table 1: Times (in minutes) for participants to complete each task.

	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 2: Tasks success rates

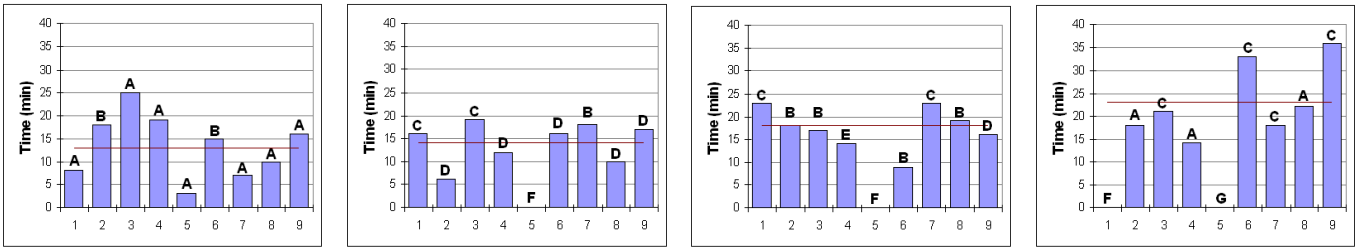
Task	1	2	3	4	Rate(%)
(A) Success no help	7			3	27.8
(B) Success minor help	2	1	4		19.4
(C) Success major help		2	2	4	22.2
(D) Minor errors no help		5	1		16.7
(E) Minor errors minor help			1		2.8
(F) Failure (major errors)		1	1	1	8.3
(G) Not attempted at all				1	2.8

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, we decided to allow struggling participants 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 2 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 (i.e. starting the guidance at the 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 2 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 3 summarizes the main topics mentioned in the questionnaire. Their answers were then discussed during the interview. In fact, the participants gave positive feedback. Table 3 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



(a) Task 1 (b) Task 2 (c) Task 3 (d) Task 4

Figure 5: Task completion times (in minutes) with the average for each task and the type of success for each participant (see also Table 2 for the meaning of the letter codes).

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 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 with 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 the availability of 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) 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.

Table 3: 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".

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

Holistically, including the interview, we conclude 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.

4.5 Discussion & Future Work

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 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 (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, now fixed, were revealed. Also, in future work, we will extend the number of interaction techniques available in the HITPROTO prototype.

5 Conclusion

This paper presented HITPROTO, a tool for prototyping haptic interactions in the context of haptic data visualization. The primary goal is to allow rapid prototyping of haptic interactions without requiring prior programming skills. The tool provides user-tunable visual programming blocks to build the interaction. We believe that learning to use the tool to prototype interactions will 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 we are focusing on under-researched areas in haptic data visualization such as tables and networks.

References

[1] 3ds Max, a 3D modelling, animation and rendering software. http://www.autodesk.co.uk/3dsmax [Last accessed: 17-12-09].

[2] Blender, "a free open source 3D content creation suite". http://www.blender.org/ [Last accessed: 17-12-09].

[3] Immersion Studio. http://www.immersion.com/products/rumble-technology/immersion-studio.html [Last accessed: 17-12-09].

[4] Rhino. http://www.rhino3d.com/ [Last accessed: 17-12-09].

[5] the H3DAPI, by SenseGraphics. http://www.h3dapi.org/ [Last accessed: 17-12-09].

[6] the X3D Standard. http://www.web3d.org/ [Last accessed: 17-12-09].

[7] C. Appert, S. Huot, P. Dragicevic, and M. Beaudouin-Lafon. FlowStates : Prototypage d'application interactives avec des flots de données et des machines à états. In *IHM*. ACM, Oct. 2009.

[8] R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments. In *CHI*, pages 537–544. ACM, 2003.

[9] W. Book. Control Prototyping with LabVIEW for Haptic and Telerobotic Systems. Webcast, March 2008.

[10] K. Bowen and M. K. O'Malley. Adaptation of Haptic Interfaces for a LabVIEW-based System Dynamics Course. In *HAPTICS*, pages 147–152. IEEE, 2006.

[11] J. De Boeck, C. Raymaekers, and K. Coninx. Are existing metaphors in virtual environments suitable for haptic interaction. In *VRIC*, pages 261–268, Apr. 2005.

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

[13] F. De Felice, G. Attolico, and A. Distante. Configurable Design of Multimodal Non Visual Interfaces for 3D VE's. In M. E. Altinsoy, U. Jekosch, and S. Brewster, editors, *HAID*, volume 5763 of *LNCS*, pages 71–80. Springer, Sept. 2009.

[14] P. Dragicevic and J.-D. Fekete. Support for Input Adaptability in the ICON Toolkit. In *ICMI*, pages 212–219. ACM, Oct. 2004.

[15] M. Eid, A. Alamri, and A. E. Saddik. MPEG-7 Description of Haptic Applications Using HAML. In *HAVE*, pages 134–139. IEEE, Nov. 2006.

[16] M. Eid, S. Andrews, A. Alamri, and A. El Saddik. HAMLAT: A HAML-Based Authoring Tool for Haptic Application Development. In M. Ferre, editor, *Eurohaptics*, volume 5024 of *LNCS*, pages 857–866. Springer, Jun. 2008.

[17] M. J. Enriquez and K. E. MacLean. The Hapticon Editor: A Tool in Support of Haptic Communication Research. In *HAPTICS*, page 356, Los Angeles, CA, USA, 2003. IEEE.

[18] P. Figueroa, M. Green, and H. J. Hoover. InTml: a Description Language for VR Applications. In *Web3D*, pages 53–58. ACM, Feb. 2002.

[19] N. Forrest and S. Wall. Protohaptic: Facilitating Rapid Interactive Prototyping of Haptic Environments. In *HAID*, volume 2, pages 21–25, Aug–Sept. 2006.

[20] S. Huot, C. Dumas, P. Dragicevic, J.-D. Fekete, and G. Hégron. The MaggLite Post-WIMP Toolkit: Draw It, Connect It and Run It. In *UIST*, pages 257–266. ACM, Oct. 2004.

[21] L. Kurmos, N. W. John, and J. C. Roberts. Integration of Haptics with Web3D using the SAI. In *Web3D*, pages 25–32. ACM, Jun. 2009.

[22] J.-Y. L. Lawson, A.-A. Al-Akkad, J. Vanderdonckt, and B. Macq. An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. In *EICS*, pages 245–254. ACM, Jul. 2009.

[23] J. Lee, J. Ryu, and S. Choi. Vibrotactile score: A score metaphor for designing vibrotactile patterns. In *WHC*, pages 302–307. IEEE, 2009.

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

[25] S. Panëels and J. C. Roberts. Haptic Guided Visualization of Line Graphs: Pilot Study. In *HAID, poster and demo proceedings*, pages 5–6, Nov. 2007.

[26] S. Panëels and J. C. Roberts. Review of Designs for Haptic Data Visualization. *Transactions on Haptics*, 2009. PrePrint.

[27] S. Panëels, J. C. Roberts, and P. J. Rodgers. Haptic Interaction Techniques for Exploring Chart Data. In M. E. Altinsoy, U. Jekosch, and S. Brewster, editors, *HAID*, volume 5763 of *LNCS*, pages 31–40. Springer, Sept. 2009.

[28] A. Ray and D. A. Bowman. Towards a System for Reusable 3D Interaction Techniques. In *VRST*, pages 187–190. ACM, 2007.

[29] J. Roberts and S. Panëels. Where are we with Haptic Visualization? In *WHC*, pages 316–323. IEEE, 2007.

[30] 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*, pages 1246–1250. IEEE, Aug. 2005.

[31] J. Rubin. *Handbook of Usability Testing: how to plan, design and conduct effective tests*. Wiley Technical Communications Library. John Wiley and Sons, Inc., April 1994.

[32] M. Serrano, L. Nigay, J.-Y. L. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef. The OpenInterface Framework: a Tool for Multimodal Interaction. In *CHI EA*, pages 3501–3506. ACM, Apr. 2008.

[33] C. Swindells, E. Maksakov, K. E. MacLean, and V. Chung. The role of prototyping tools for haptic behavior design. In *HAPTICS*. IEEE, 2006.

[34] Y. Visell, A. Law, and J. R. Cooperstock. Toward iconic vibrotactile information display using floor surfaces. In *WHC*, pages 267–272. IEEE, Mar. 2009.