A Framework for the Development of Haptic-Enhanced Web Applications

Sara Comai, Davide Mazza, and Andrea Guarinoni

Politecnico di Milano
Department of Electronics and Information (DEI)
Piazza L. Da Vinci 32,
I-20133 Milan, Italy
{sara.comai,davide.mazza}@polimi.it

Abstract. In the last years we have witnessed an increasing adoption of haptic devices (allowing the user to feel forces or vibrations) in several fields of applications, from gaming, to mobile, automotive, etc. Some efforts have been done to enhance also Web applications interfaces with haptics, either to improve accessibility or, more in general, to improve usability. Despite the spreading of haptic applications, their development is still a time consuming task that requires significant programming skills. In particular, in the Web context no plug-ins or style extensions are currently available and applications must be developed from scratch. In this paper we describe a framework to easily include haptic interaction in Web applications, focusing both on haptic interaction modeling and on its implementation.

1 Introduction

Haptics, the technology that exploits the human sense of touch by applying forces, vibrations, or motions to user's hands or body, has received an enormous attention in the last decades, but only in recent years has reached an important level of visibility in several fields of applications. The most spread form of haptics is represented by the tactile effects on mobile devices, where the user can feel vibrations through the skin; similar effects can be found also in the automotive field, where touch surfaces are replacing mechanical buttons, as well as on cameras and media players' touchscreens. Haptics has been largely exploited also in the gaming field, where controllers like the DualShock^{TM} by Sony or the Falcon^{TM} by Novint provide stronger vibrations with different levels of intensities according to the action of the game.

But vibrations are not the only feedback that can be provided by haptic devices. More sophisticated input/output devices like, e.g., the PHANTOM^{TM} stylus by Sensable, can provide richer effects, by supporting a bidirectional communication of the forces: the user can *feed* the system with data about his/her movements (in terms of position, velocity of a movement, or even direct force supplied through the device) and can *receive* information from the system in the form of sensations simulating weight, resistance, etc. Such devices have been

M. Brambilla, T. Tokuda, and R. Tolksdorf (Eds.): ICWE 2012, LNCS 7387, pp. 253–267, 2012.

traditionally employed to improve virtual reality systems and simulators, but different works show how to use them also in the context of Web applications [7, 12–14].

Haptic-enhanced Web applications offer new modalities of interactions and new possible scenarios. Beside navigating the Web by associating events with vibrations in a way similar to what mobile devices provide (e.g., to confirm user's inputs or as non-intrusive alerts when calls/messages arrive), force feedback devices such as mice, joysticks, pens, etc. can supply richer kinesthetic forces producing moderate attractive/repulsive effects, so that the following use scenarios can be devised [7, 13, 15]:

- when the user presses a button, the visual aspect of the button may change and the user may "feel" the simulation of "mechanical" pressure, thus providing an immediate and more realistic confirmation of user's action;
- while the user reads the main content of the page, he can "feel" the scrollbar without the need of looking at it, thus reducing his/her workload and avoiding distractions from his/her main task;
- when the cursor is over a button or a form field, the user feels a "snap-to" effect that avoids slip-offs, thus reducing possibilities of errors and guaranteeing a more precise execution of the task;
- when the user accesses a complex page, with rich content, haptic cues can highlight or indicate the importance of user actions or of the displayed information (e.g., recency of news), without overloading the visual channel; or when ads or auxiliary content can lead to visual distractions, a gentle guided movement of the cursor towards core contents can be provided.

Experimental works show that the integration of haptics in Web applications is helpful, in particular, in terms of reduced workload and reduced number of errors [15]; moreover, in case of applications targeted to visually impaired users, also accessibility can be improved [12, 14].

However, in the current practice, the development of haptic applications is still a time consuming task and requires significant programming skills. The solutions proposed in literature for adding haptic interactions into Web pages are mainly hand-crafted and specifically tailored to the needs of each work: the field would surely benefit of a more structured approach to be integrated into existing methodologies and tools, according to the usual Web engineering principles.

At this aim, we propose a framework to include haptic interaction in HTML-based Web applications, supporting a flexible design of haptic effects and separating device-independent features to be easily extendable to different haptic devices. After an initial overview in Section 2 of the related works proposed in literature, both on the introduction of haptic interactions in the Web and on the current practice for the development of haptic applications, Section 3 introduces the basic concepts underlying haptic interactions. Section 4 presents the design of our framework, while Section 5 describes an associated tool to ease the development of haptic-enhanced applications. Section 6 analyzes the impact

of adding haptic interaction in the usual Web browsing activity and, finally, in Section 7 conclusions are drawn.

2 Related Work

The research in the last decade has shown an increasing trend of interest in the adoption of haptics in common applications. However, in order to be applied in the Web context, several issues need still to be investigated.

The problem of how to interact with the Web through a haptic device has been first treated in [6, 11, 19]. [11] tries to introduce haptic modalities for the *exploration* of Web pages, by assigning forces or vibrating effects to the different widgets of a page (e.g., textfields, buttons, images). The proposed approach is targeted to people with visual disabilities and the original Web page is mapped into a 3D virtual environment, where each widget of the original Web page is represented by a typed 3D object providing both haptic and audio feedbacks (e.g., texts are mapped into "T" objects having a particular shape and friction, hyperlinks into "arrow" objects having a different shape and surface characteristics, and so on). Also [19] aims at improving accessibility on the Web by means of a "content-aware" Web browser plug-in, where audio and haptic modalities are supported: the user can be informed when (s)he is in the vicinity of an image or an hyperlink, to increase the level of spatial awareness, and can "feel" the different types of objects by means of effects associated with users' actions like hovering a link, an image and so on. In a similar way, [6] utilizes haptic feedback and speech recognition to aid browsing, but with a limited set of haptic effects. Also maps, images, and other graphical contents can be haptically rendered with groove, texture, or vibration effects so that the user can "feel" the pictures: for example, city maps can be explored by following groove lines representing the streets [12]; charts can be associated with haptic effects proportional to the quantitative information they represent [16]. Effects have been applied effectively also to better locate objects in the space of interaction [15] or to be directed to a specific target (e.g., after filling a search box the user is guided to the submit button) [13], according to the user's task.

While the design of haptic interactions and their inclusion in Web applications have been studied by different researchers, the development methodology used for haptic applications is still a critical aspect in this field. In general, the current practice in the development of software with haptic interfaces consists mainly in writing the corresponding code manually by including API libraries or SDK provided by the device manufacturer.

Commercial libraries depend on the device itself: for example, for the Sensable PHANTOM [5] the HDAPI / HLAPI [10] are available. The different APIs are incompatible, but some efforts have been done to realize *device-independent* libraries that try to integrate the features supported by different haptic devices. The most known examples are represented by the Haptik library [17] and by the library provided by Novint [4], which offer *Hardware Abstraction Layers* for accessing the device features.

Among the academic proposals, CHAI3D [1] is a set of open-source libraries to be used for the development of applications with the C++ language. The APIs focus primarily on the management of scene graphs (through OpenGL) and on integrating haptic effects into a 3D scene. JTouchToolkit [3] is another open source library that can be used to introduce haptic interactions in a Java environment.

At a higher level of abstraction, [9] proposes HAML, an XML-based specification language that couples applications with haptic devices, defining the hardware properties of the haptic tool and modeling the haptic response of objects' surfaces in terms of stiffness, resistance, and so on. The HAML formalism is completely textual and is supported by the authoring tool HAMLAT [8], which allows users to render the haptical properties of a virtual environment with no programming skills. However, all the proposed libraries and specification languages are very general and address mainly hardware aspects, and/or are typically conceived for 3D graphics.

3 A Model for Haptic Interaction in Web Applications

The elements of the graphical interface of a Web application are usually interpreted as passive components: the user decides to interact with them according to his/her own will or the task (s)he would like to perform. However, each element can become *active*, by adding proper haptic effects that may guide the user during the navigation.

Systems involving haptic interactions are typically based on the architecture shown in Figure 1 [18], which includes:

- The *haptic device*, providing in input to the system the position of its proxy, and rendering in output a force;
- The *video* rendering the visual aspects of the application;
- A module devoted to the *haptic rendering*;
- A module devoted to the *visual rendering*;
- A module containing the *objects of the environment* to be rendered visually and associated with haptic effects.

The haptic rendering pipeline is composed of three main blocks:

- *Collision detection*: given the current position of the haptic proxy, it determines which virtual objects collide with the proxy;
- *Force response*: the interaction force between the proxy and the virtual objects is computed. Force computation is based on the *mathematical* representation (model) of the force, that is a function of the input provided to the device (typically, the position of the haptic proxy, but also its velocity and acceleration) and, possibly, of the features of the object associated with the haptic behavior. The mathematical model is used to compute the intensity and the direction of the force to render: the output may be perceived by the user as a change of the position of the proxy, possibly with a given velocity and acceleration.

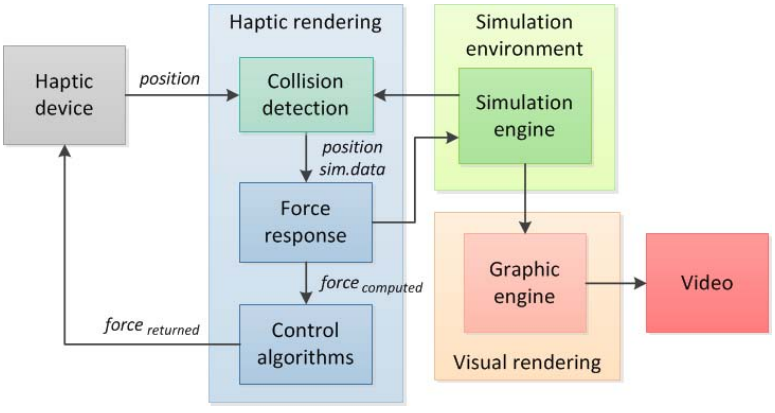

Fig. 1. Basic architecture for haptic rendering

– *Control algorithm*: the computed force is approximated according to the device's capabilities (e.g., maximum output force) and rendered to the user.

Considering Web applications, any element of the page may be associated with an effect that the user will perceive during navigation. From a conceptual point of view we need to identify such elements and to understand when and how forces need to be computed and rendered. At this aim, haptic interaction can be modeled through the specification of a set of tuples of the type $\langle obj, event, effect, action \rangle$ where:

– *obj* is the *target object*/widget of the application associated with a haptic effect; in a HTML page it may be any object registered inside the Document Object Model (DOM) of the page (from a *HTMLDivElement* to a *HTMLInputElement* or *HTMLDocumentElement* etc.);
– *event* is the *event* associated to *obj*: it may be a native event of the browser (like click, load, mouseover, etc.) or an application-dependent event defined by the developer[1];
– $effect$ is the *effect* generated by the haptic device and felt by the user. Examples of effects include: vibration effects or attraction/repulsion forces. They are specified by means of mathematical models, as exemplified next. Effects may be:
  • *Single*: they are defined by means of a single (possibly complex) expression. Single effects are associated with a *timeout* specifying a time duration (e.g., 1000 ms);
  • A *Concatenation* of effects (Eff_1, Eff_2, ..., Eff_n), where each effect may be in turn either a single effect or a concatenation of effects: in this case, one effect a time is rendered, and the whole sequence may be associated with a number of *iterations*, to be possibly repeated several times.

[1] Application-dependent events may be defined by means of the *Object.createEvent()* function, according to the ECMAScript 5 specifications.

Force model	Description	Parameters	Input	Expression
Spring 	Elastic force depending on the position of the proxy	Elastic constant ($\vec{k_e}$) Point Of Interest ($\overrightarrow{POI}$)	Position ($\vec{x}$)	$\vec{F} = \vec{k_e} * (\vec{x} - \overrightarrow{POI})$
Magnet 	Magnetic force depending on the position of the proxy	Max attraction ($\vec{k}$) Point Of Interest ($\overrightarrow{POI}$) Gradient ($\vec{\sigma}$)	Position ($\vec{x}$)	$\vec{F} = \vec{k}/(\vec{x} - \overrightarrow{POI})^{\vec{\sigma}}$
Damper 	Damping force depending on the velocity of the proxy	Viscous constant ($\vec{k_v}$)	Velocity ($\vec{v}$)	$\vec{F} = \vec{k_v} * \vec{v}$
Mass 	Force depending on the acceleration of the proxy	Virtual mass (m)	Acceleration ($\vec{a}$)	$\vec{F} = m * \vec{a}$
Vibration 	Vibration force generated by a sinusoidal oscillation	Amplitude ($\vec{A}$) Angular frequency ($\vec{\omega}$)	Time elapsed (t)	$\vec{F} = \vec{A} * \sin(\vec{\omega} * t)$
NullModel 	Idle force returning a null value	---	---	$\vec{F} = 0$

Fig. 2. Examples of force models

– *action* specifies if, upon the event, the associated haptic effect must be *started*, *stopped*, or definitely *removed* from the page. Stopped effects can be restarted when the event associated to the object occurs again; instead, removed effects can occur only once for the loaded page. Given an effect, stop/remove actions may be not specified, since timeouts and iterations are enough to stop the rendering of the effect; when both are specified, the effect will cease upon the occurrence of the first blocking event (timeout or stop/remove).

This model extends our previous proposal done in [7] by taking into account the specific field of application. In our work we have implemented some predefined effects. Examples of such effects are reported in Figure 2: each effect is specified by means of a mathematical expression that depends on inputs received from the device (e.g., the position of the proxy) and possibly on other parameters.

Figure 3 shows an example of Web page enhanced with haptic effects applied to the HTML elements of the Google homepage. Due to the lack of guidelines for introducing haptic effects to enhance user interaction, we enriched the visual information with haptic feedbacks conveying messages not already communicated

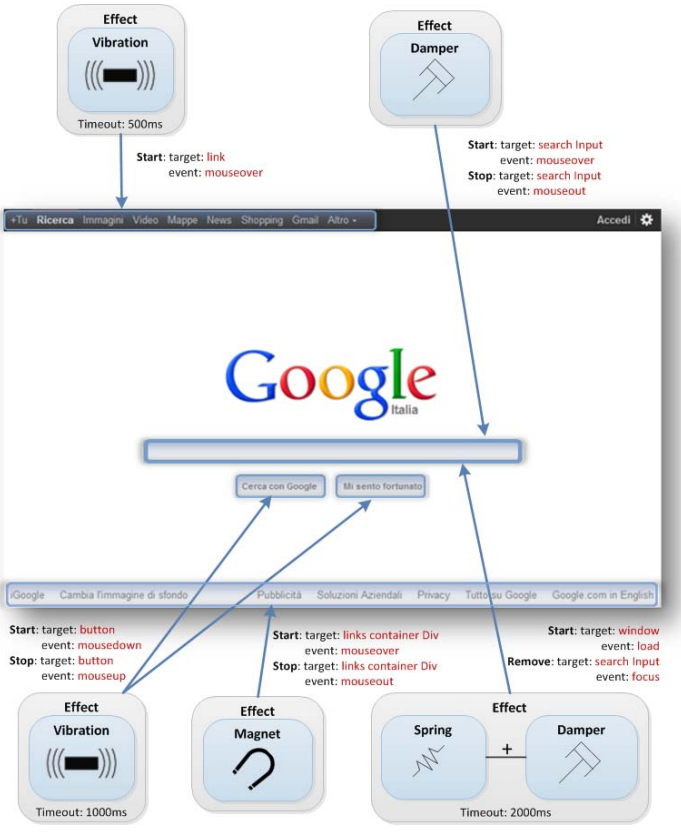

Fig. 3. An example of page enriched with haptic effects

to the user. For example, we assigned a time-dependent vibration perceivable by the user as soon as (s)he moves the mouse over the individual links listed in the upper part of the page, or over the buttons near the search bar, to confirm user's action (typically not emphasized in a visual way). Moreover, in order to assist the user during the navigation, the menu items at the bottom of the page are associated with a slight magnetic attraction to improve the cursor control and avoid mouse slippery on click. In the same way, a gentle guided movement over the input field has been obtained as a composition of the spring and the damper force models; this effect is rendered as soon as the loading process of the page is completed, while a viscous (damper) effect is perceived whenever the cursor moves over the input field. The playback of each single haptic effect is regulated by means of the associated events/actions, which specify the execution criteria of the handlers that activate and deactivate the effect. For example, on the two buttons in the middle of the page, the haptic effect will start whenever the *mousedown* event occurs and will stop either after 1000 ms or before, if a *mouseup* event is fired.

4 A Framework for the Addition of Haptic Effects in Web Applications

The framework proposed in this paper is based on the architecture depicted in Figure 4, which contextualizes the classical schema of Figure 1 for the Web realm. The Web page is extended with Javascript code exploiting a Javascript library (called *JHaptic*) that is responsible for the computation of the force and its scaling (i.e., of the Force Response and Control Algorithm steps of the haptic rendering pipeline). A browser plug-in has been developed to enable the interaction with the physical device through the APIs exposed by it (see Section 4.2).

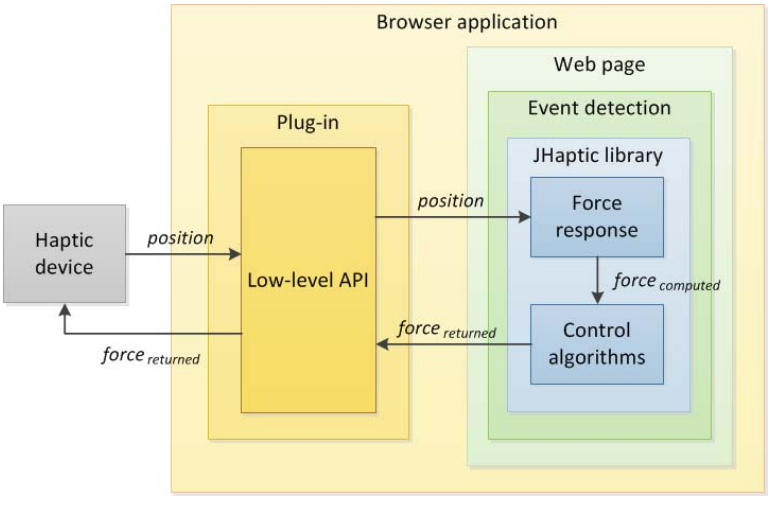

Fig. 4. The working architecture of the proposed library

Compared to the classical haptic rendering pipeline, the collision detection step can be obtained by exploiting the event model of the browser rendering engine itself. Indeed, it is possible to monitor events like *mouseover* or *mouseout*, over any element (visible or not) stored in the Document Object Model (DOM).

The communication among the different modules depicted in the figure is carried out as follows:

1. The current position of the haptic interface is taken in input; this is processed by the low-level libraries provided with the selected device and interfaced through a plug-in with the Web page loaded in the browser;
2. A Javascript library is integrated within the HTML document; it receives the position of the device proxy, maps it to the corresponding point both in the desktop and in the Web page and computes the forces to be reproduced in output; effects are included in the Web application code by invoking the corresponding library functions;

3. The value of the computed force is scaled so that the hardware limits of the adopted device are not beyond its force magnitude rendering capabilities;
4. By exploiting the low-level APIs through the plug-in, the computed force impulse is sent to the device, so that it is perceived by the user.

The library therefore manages the whole rendering process and exploits an external plug-in (a Dynamic-link library) to access the APIs needed to interact with the physical device.

In order to allow Web contents browsing, the mouse cursor has been firmly associated to the end-effector of the haptic device so that a movement of it produces a proportional shift in the position of the mouse arrow on the screen. This way, the user can do actions and fire events as with the usual mice, and haptic effects ca be rendered (started, stopped, etc.) upon a particular occurring condition (e.g., click on a button, hovering a link, missing data in an input field, etc.) of a given element (or node) of the page.

The framework can supports any kind of device, from (2D) mice to haptic devices moving in a 3D working space: in the latter case, the position of the haptic proxy must be mapped into the bi-dimensional position of the pointer on the screen; in our model, the user screen space has been vertically centered onto the workspace of the physical device, represented as a parallelepiped. The currently displayed area of the Web page (*viewport*) is conceived as a vertical flat surface, so that the three-dimensional position of the end-effector is projected on the vertical plane in which the Web page resides, thus providing the bi-dimensional coordinates of the mouse on the screen (see the Virtual I/O panel in Figure 6).

4.1 JHaptic Library

The conceptual elements introduced in Section 3 to model haptic interaction inside a Web page have been integrated into the JHaptic library according to the UML diagram in Figure 5. It represents only the most significative objects composing the project and their fundamental properties.

The library presents some first-class objects to model the haptic interaction. The *Device* class models the features offered by a generic haptic device like its inputs (*position*, *speed*, and *acceleration*) and ouptut (*output force*). The size of the physical workspace and the intensity of the maximum playable force are obtained directly from the parameters exposed by the plug-in, allowing the library to abstract from the device type.

A *Virtual Device* is also defined emulating the functionalities of a basic haptic device, so that the testing of the application is possible also without the need of a connected physical device (see Section 5).

The *ForceModel* class, and in particular its *func* property, represents the mathematical model of the effect to render. To consider the most general working space of haptic devices, all input and output data are represented by 3D vectors (class *Vector*). Some widely-used models have been predefined and represented by utility classes such as the *Spring*, *Damper*, or *Vibration* force models, etc.,

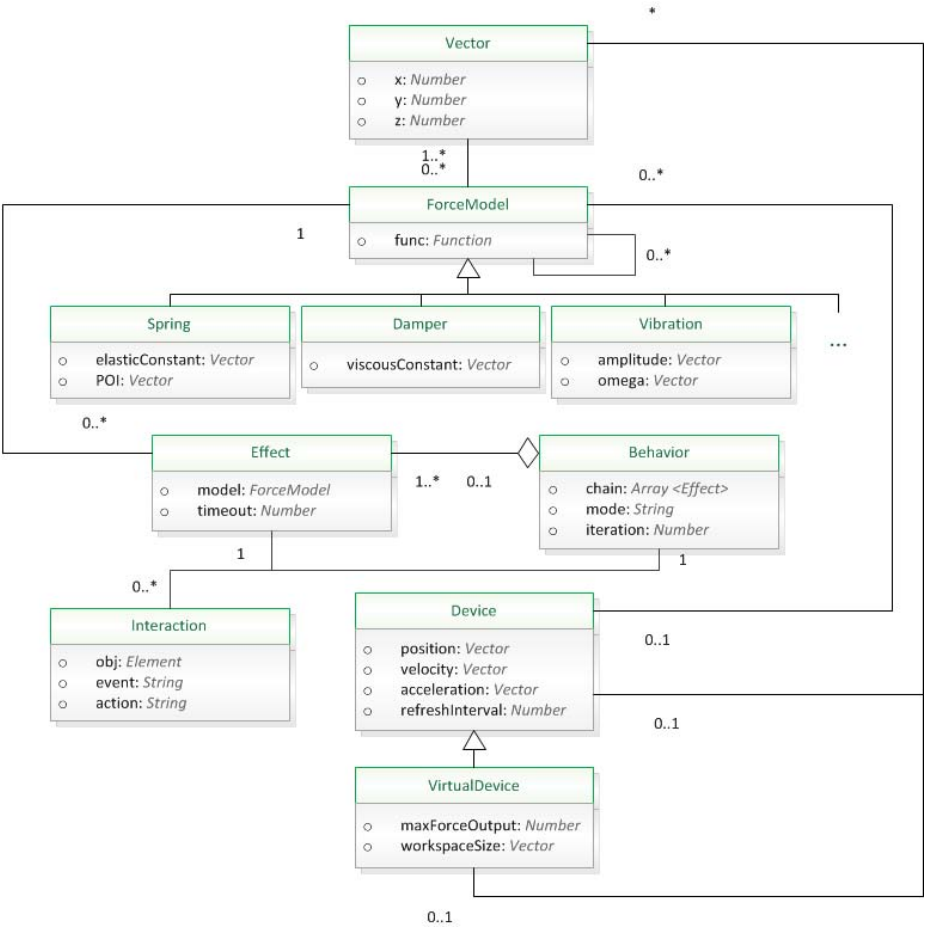

Fig. 5. The UML model of the developed library

just to mention the ones depicted in Figure 5. Anyway, the developer can customize the effects to render to face user's needs: the class *Effect* allows to specify a duration in time of the force feedback generated by the defined model; moreover, effects can also be composed, in order to obtain a time-sequenced effects or specifically designed patterns: this effects composition is possible with the *Behavior* class which supports the concatenation of effects, to reproduce them for a specified number of times.

4.2 The Plug-in for the Browser

The developed plug-in has been designed to be used with NPAPI-compliant browsers, e.g., Mozilla Firefox, Google Chrome, Gecko, Safari, etc. Among all the browsers, we have considered Google Chrome and for the development of the

plug-in we have used Firebreath [2], a framework that allows the creation and development of simple plug-ins for most of the popular browsers integrating the NPAPI functionalities.

The plug-in has been implemented taking into account two main requirements:

- *Low complexity*: the minimal functions needed to manage the communication with the device are included. In this way, the creation of new plug-ins to extend the library support to more devices becomes simpler and faster, since no algorithms for the computation and management of the haptic rendering need to be redefined.
- *High compatibility*: the APIs exhibited by the plug-in should represent the basic set of functions needed to interface with a haptic device. The plug-in is the main interface for the JHaptic library: if the plug-in exhibits a fixed set of APIs, compatible with all the devices, this will simplify a possible change of the employed haptic device.

To maximize the compatibility with more devices, 3 DOFs (degrees of freedom) have been considered. At this aim a basic set of methods and variables that the plug-in should exhibit to the JHaptic library has been identified, including:

- a set of methods to control the device, i.e., to start and stop the communication with the device;
- a method to send the force value to be instantaneously rendered;
- a method to obtain the force applied to the device;
- a method to obtain the current coordinates of the proxy;
- a method to set the working space (2D or 3D) and a variable to set its maximum dimensions;
- a variable to set the maximum force value that can be rendered by the device.

For further details the reader can refer to the official Web site of the JHaptic library project[2], where the complete framework (library and plug-in) can be downloaded, more specific documentation can be found, and examples of Web pages enriched with haptic effects are available.

In our tests we have used the Sensable PHANTOM^{TM}, which is a 6-DOF device: the additional 3 DOFs exhibited by the end-effector of this specific device have not been considered.

In order to decouple the implementation from a specific device, we have not used any proprietary library associated with a particular haptic interface, but we have written the plug-in code with the help of the multi-device CHAI3D open-source library [1], which allows the automatic detection and adaptation of the plug-in functionalities to other haptic device.

The plug-in takes care also of the movements of the mouse pointer on the screen according to the movements of the haptic end-effector, by mapping the corresponding coordinates as explained in Section 4. To handle the positioning of the mouse cursor at the computed screen coordinates, appropriate operating system APIs are called from the plug-in, since this is not possible using Javascript code.

[2] `http://home.dei.polimi.it/mazza/jhaptic`

Device-Independency. A note on the device-independency of the overall design is here worth saying. The whole framework has been designed to be device-independent: the Javascript library is based on a conceptual model specifying the haptic interaction, and also the plug-in has been implemented using a device-independent library such as CHAI3D and has been designed considering 3-DOF devices, so that it can be easily adapted to be used with different physical interface: in case of 2-DOF devices the z-coordinate of the Vector is simply ignored; for devices with more than 3 DOFs, only the main 3 DOFs are considered.

5 Console for Debugging and Testing

In order to support the developer in the usage of the proposed library we realized a browser-integrated tool to simplify the debugging and testing process of a Web page extended with haptic feedback. The solution provided is made up of a console and a simulator. Both components are implemented in Javascript so that they can be exploited directly within the browser, thus overcoming the problems related to the architecture or the operative system of the developer's machine. The structure of the debugging tool is shown in Figure 6.

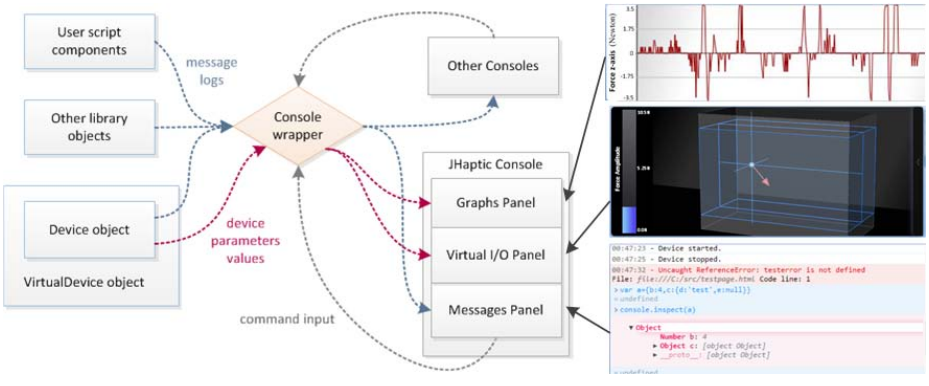

Fig. 6. The structure of the debugging tool

By means of the console, injectable on request within the HTML document, the developer is able to visually monitor the behavior of the haptic device by exploiting three panels:

1. The *Graphs* panel, displaying a chronological visualization of the changes occurring during haptic interaction in terms of wave forms representing the force pulses reproduced by the device; it is also possible to visualize the variations of the other measures computed during the haptic rendering process (like position, velocity, refresh rate, etc.) by means of multiples real-time quoted plots;

2. The *Virtual I/O* panel, graphically rendering in the 3D workspace the representation of the current position of the proxy and the magnitude and direction of the output force, along with other auxiliary information such as the arrangement of the desktop area or the virtual placement of the page viewport plane;
3. The *Messages* panel, used to monitor the log, warning and error messages related to the functioning of the Web page and of the library components.

Through the console it is also possible to directly edit at runtime the haptic effects set in the Web page, taking advantage of the code injecting feature that has been developed.

All the information and functionalities provided by the console are accessible either using a real device, controlled by means of a browser plug-in, or using a simulated one, to allow the test of an application without the need of a physical device connected to the system. At this aim, the simulator emulates the three-dimensional input received from a real device by conveying the correct position calculated by parsing the events related to the mouse positions and buttons, according to the workspace size defined by the developer; the output is shown in the console.

6 Evaluation and Experience

In order to understand the impact of the introduction of haptic interaction on Web navigation, we have done some performance tests on different hardware configurations by extending different pages with multiple haptic feedbacks. We evaluated more usage conditions for the library, exploiting both the simulator and the real device used through the developed plug-in. The tests were performed on mid-range systems: 1) CPU Intel Core 2 Duo T6500 2.1GHz, RAM 4GB DDR2 800MHz, GPU Ati Mobility Radeon 4650; 2) CPU Intel Core 2 Duo E8400 3GHz, RAM 4GB DDR2 800MHz, GPU nVidia Geforce 9600GT; the Google Chrome (v.17) browser on the Microsoft Windows 7 32-bit operative system has been employed. The following parameters have been monitored: 1) the *haptic refresh rate*, i.e., the real frequency with which the force pulses are computed and sent to the device 2) the *CPU workload*, i.e., the processing power required to perform the calculation of the forces set, keeping the ideal refresh rate to 1 kHz. Results show that the rendering of a low number of haptic effects within the HTML document does not lead to a significant increase of the resources usage. In particular, the playback of a single force model among those already provided in the library does not cause a tangible variation of the CPU workload, while the real refresh rate remains close to 1 kHz. By increasing the amount of the haptic effects simultaneously rendered, we observed a slight linear rise of the CPU power demand, whereas the average haptic refresh on the low-end system showed more frequent peaks of delay. Anyway, several force models to be reproduced simultaneously within a Web page is a situation that will hardly occur in practice: user studies presented in literature demonstrated that the design of the Web page should enable one - possibly complex - effect at a time,

to avoid a chaotic sensations to the user receiving too many information from different channels [13]. Moreover, the CPU usage is never overloaded, allowing the execution of other applications in parallel to the Web browsing activity. Finally, no performance difference has been identified in the use of the real and the simulated device.

On the development experience side, the framework presents mainly two benefits: 1) compared to manual coding, the JHaptic library, together with the plug-in interfacing the physical device, simplifies the addition of the haptic effects to a Web application; 2) moreover, the framework can be extended with any new effect, by means of additional Javascript function(s) that model the effects to render by exploiting the already existing JHaptic library APIs; extensibility is an important aspect, especially when common standards are lacking, like in the case of haptics.

7 Conclusions and Future Works

In this work we have designed and realized a framework for the introduction of haptic interactions into Web applications. To our knowledge, this is the first framework supporting a flexible design of effects, providing a set of browser-integrated tools to help the Web developer in the definition and testing of haptic effects, and designed to be device-independent.

As future work we plan to extend the capabilities of the developed library with the integration of a graphical engine, for the specification of haptic effects within a three-dimensional environments rendered in the browser. Another aspect worth studying concerns the usability of haptic-enhanced Web applications, considering the type of device, the context of use, and the type of target users, in order to identify guidelines and design patterns.

More room there will be in the future for haptic technologies: the research trend emerged in recent years pays particular attention to the design and development of devices exploiting the sense of touch, also the W3C, has set up two relevant initiatives[3] to face the emerging new ways of interactions and the supporting devices. Haptics is going to strongly influence consumers' habits and research domains for the next years.

References

1. Chai 3d, `http://www.chai3d.org`
2. Firebreath, `http://www.firebreath.org`
3. Jtouchtoolkit, `https://jtouchtoolkit.dev.java.net/`
4. Novint, `http://home.novint.com/`
5. Sensable phantom omni, `http://www.sensable.com/haptic-phantom-omni.htm`
6. Caffrey, A., Mccrindle, R.: Mccrindle r. developing a multi-modal web application. In: Proceedings of ICDVRAT 2004, pp. 165–172 (2004)

[3] The Device APIs Working Group: `http://www.w3.org/2009/dap/`
The Vibration API: `http://www.w3.org/TR/2012/WD-vibration-20120202/`

7. Comai, S., Mazza, D.: Introducing haptic interactions in web application modeling. In: WSE, pp. 43–52 (2010)
8. Eid, M., Andrews, S., Alamri, A., El Saddik, A.: HAMLAT: A HAML-Based Authoring Tool for Haptic Application Development. In: Ferre, M. (ed.) EuroHaptics 2008. LNCS, vol. 5024, pp. 857–866. Springer, Heidelberg (2008)
9. Eid, M., Mansour, M., Iglesias, R., Saddik, A.E.: A device independent haptic player. In: Proceedings of the 2007 IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems. IEEE (2007)
10. Itkowitz, B., Handley, J., Zhu, W.: The openhaptics toolkit: A library for adding 3d touch navigation and haptics to graphics applications. In: Proceedings of First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC 2005). IEEE (March 2005)
11. Kaklanis, N., Calleros, J.G., Vanderdonckt, J., Tzovaras, D.: A haptic rendering engine of web pages for blind users. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2008, pp. 437–440 (2008)
12. Kaklanis, N., Votis, K., Moschonas, P., Tzovaras, D.: Hapticriamaps: towards interactive exploration of web world maps for the visually impaired. In: W4A, p. 20 (2011)
13. Kuber, R., Yu, W., McAllister, G.: Towards developing assistive haptic feedback for visually impaired internet users. In: CHI, pp. 1525–1534 (2007)
14. Kuber, R., Yu, W., O'Modhrain, M.S.: Evaluation of haptic html mappings derived from a novel methodology. TACCESS 3(4), 12 (2011)
15. Oakley, I., McGee, M.R., Brewster, S.A., Gray, P.D.: Putting the feel in look and feel. In: CHI, pp. 415–422 (2000)
16. Panëels, S.A., Roberts, J.C.: Review of designs for haptic data visualization. IEEE T. Haptics 3(2), 119–137 (2010)
17. Pascale, M.D., Prattichizzo, D.: The haptik library. IEEE Robotics & Automation Magazine 14(4), 64–75 (2007)
18. Salisbury, K., Conti, F., Barbagli, F.: Haptic rendering: Introductory concepts. IEEE Comput. Graph. Appl. 24(2), 24–32 (2004)
19. Yu, W., Kuber, R., Murphy, E., Strain, P., McAllister, G.: A novel multimodal interface for improving visually impaired people's web accessibility. Virtual Reality 9(2-3), 133–148 (2006)