An Audio Haptic Tool for Visually Impaired Web Users

Catherine A. Todd, Ammara Rounaq, Umm Kulsum Nur, Fadi Boufarhat

An application that functions as a browser has been created for visually impaired Web users. It is distinctive from popular browsers such as Internet Explorer, Firefox and Google Chrome as it is designed specifically for partially sighted and blind Web users. The custom browser provides the feel of website elements via sense of touch (Haptic feedback), combined with speech synthesis. Force and sound feedback are provided through a virtual reality environment containing representations of 3D Web elements from the accessed website. This paper presents preliminary findings from the browser design and implementation of Haptic and audio Web element representation. Future work aims at creating a complete 3D Web browser with an interactive GUI that supports various features of existing browsers such as History, Favorites, Help and Image data.

Keywords: Web browser, Haptic feedback, Visually Impaired, GUI design

1. INTRODUCTION

As reported by the World Health Organization [1], 285 million people are visually impaired and must live with partial sightedness or blindness. The majority of those suffering from visual impairment, around 90%, live in developing countries [1]. Activity fulfilment in daily life for those with such impairment requires adaptation of behavior and assistive technologies are often employed for completion of routine tasks. Custom tools such as screen readers and Braille readers facilitate computer usage such as for sending emails, reading the newspaper or browsing the Web. Browsers that provide aural feedback for users read the Web content but their functionality is limited to a few HTML elements [2] and these provide feedback in a limited, sequential manner. Users tend to lose sense of direction in the virtual space as they are unable to recognize the position of the cursor during Web navigation [2].

The Internet has become a predominant source of information for a vast global population, however its benefits to those with visual impairment are limited by the available assistive technology such as screen readers and voice browsers. As more web-based content is becoming increasingly visual in nature (including video and image-based content), WebPages are becoming more difficult for visually impaired users to access, navigate and interpret. The problems experienced by visually impaired users when using a screen reader can range from frustration caused by wasted time and effort, asking sighted people for help in understanding content and in the process of navigation, and/or abandoning the task altogether [3]. Screen readers require extensive training for effective use and lack the functionality to interpret graphical information. Interaction via screen readers limit the overall web browsing experience for since they assist navigation only in a linear and therefore, time consuming manner.

The way in which a website is designed also plays a fundamental role in user accessibility for content retrieval. Despite government legislations and guidelines to ensure wider accessibility, websites are designed with attractive visual layout without consideration of the needs of users with disabilities. Web pages contain ambiguous links, graphics without a title or description, and tables with unstructured layout; which make it difficult for those with visual impairment to understand page content or attain an overview, even with assistive technology [4]. From an implementation perspective it is impractical to develop websites specifically targeting visually impaired users as it excludes existing websites and is cost as well as time inhibitive. Therefore, different approaches need to be investigated to represent existing websites in an alternative form that offer greater access to those with disabilities in consideration of current user habits.

Visually impaired users interact with a computer mainly through the keyboard, with the aid of text to speech tools and the use of a Braille interface. Recent developments have been made in utilizing Haptic devices for content-retrieval [2] and 3D surround sound, but they have not yet proven to be beneficial [5]. Inhibitors to success include the high cost of Haptic devices, limited force sensitivity, degrees of freedom (DoF) and amplitude range and hence the subsequent limitation of the number of distinct Haptic sensations, as well as the lack of awareness of these devices and their potential utility within the visually impaired community [6].

The purpose of this work was to develop a multimodal interface that allows visually impaired persons to have enhanced access to the World Wide Web. The system aims to improve current software features and address deficiencies in the existing research by effectively combining audio and Haptic feedback for user navigation and understanding of web-based content. To enable
navigation, combined audio/Haptic feedback is used to assist visually impaired users to maintain a mental layout of a webpage as they access and interact with webpage content. The multimodal interface seeks to enhance the user’s own orientation on a webpage and locate and retrieve information easily. The main target population of users involves those with total vision impairment and those who are partially sighted.

The application works by mapping a webpage to a 3D Virtual Reality (VR) environment wherein HTML elements are visualized as 3D objects within the VR, using an open source solution. The position of the 3D objects in the VR correspond to the position of the same HTML components in the webpage. Once the HTML document is retrieved, it is transformed into an XHTML document so that it can be parsed as an XML file; the HTML document is loaded internally and parsing to extract only defined valid HTML tags. Parsing and extracting occurs concurrently, hence defining tags to be extracted makes the process fast. The position of each widget on the scene has been maintained with respect to its original position on the web, as opposed to that in [2]. In addition, various image processing techniques are implemented to add depth to the image, including edge and depth detection algorithms to convert it into 3D. A speech synthesis engine is integrated into the application, which converts text associated with webpage components into speech. For instance, a 3D object representing an image is not meaningful for a visually impaired user alone, but a description (alternate text) adds meaning.

As the 3D objects are being rendered in the VR scene, the Haptic information associated with each widget is transformed and the user may then interact with these widgets using the Haptic device. Each of these widgets has a 3D shape which differs from their website representation. The aim was to uniquely identify each shape which a visually impaired user would easily recognize using the Haptic device. Each 3D object has Haptic properties including static friction, dynamic friction, stiffness and other force effects such as pertaining to inertia, buzzing or constraint-based interactions. A low cost Haptic device, the Novint Falcon, is used for a wider user accessibility.

2. LITERATURE REVIEW

Currently, the most preferred mode of assistive technologies for accessing web-based content by visually impaired persons is a screen reader, with popular readers including JAWS (Freedom Scientific), Window Eyes (GW Micro), Web Anywhere (University of Washington) and Thunder (ScreenReader.net). A screen reader converts the text within a webpage into audio feedback (speech), which is delivered sequentially from left to right across the page, resulting in omission of important structural information. This makes it difficult for users to gain a thorough understanding of the material as well as access a particular section of text within the page. Buzzi et al [3] describe several issues that occur when interacting with web content via a screen reader: content that is read sequentially can be time consuming and redundant, especially if there is repetition between pages. Further, buttons, links and images are announced as they appear in the code which requires significant effort from the reader to interpret them as part of the interface [3]. Reading information from tables row-wise can put information contained within it out of sequence. Lack of an overview of the interface can lead the user to navigate for a long time before they find relevant information [3]. Users find it difficult to work with form elements and sometimes no alternative description is provided for non-visual content [3].

Despite the relatively high cost of Haptic devices compared to other software interfacing tools, the limited type of commercially available Haptic devices and awareness-related limitations of the technology, there is active, recent research on the use of audio and Haptic tools for the visually impaired to interact with the Internet and web-based applications. Roth et al [7] developed an audio-haptic browser for the visually impaired wherein various sounds were synthesized to represent the nature of HTML elements present on a page as they were selected, providing the user with improved awareness of their position on a page and the meaning of the element. The system used sophisticated and expensive commercially available audio and haptic devices and as such, the solution was not practical for a vast majority of web users. Recently, more feasible (inexpensive) haptic technologies are available for desktop solutions, including the Novint Falcon (Novint), Phantom Omni (SensAble Technologies) and Logitech Wingman Force Feedback Mouse (Logitech).

A multimodal application was developed by IFeel Pixel that creates audio-tactile sensations based on the structure of an image and the pixels within it as detected by the haptic device. Such multimodal applications have the ability to create meaningful non-visual representations of objects and engage the user [8]. Retard et al [9] have also used tactile feedback to represent non-visual information such as images, frames and tables allowing visually impaired users to get a basic understanding of the visual representation of a web page. Web pages are rendered using a tactile graphics display (as opposed to the use of a haptic device) and text to speech is enabled, where passages are read aloud to the user and zooming in on text and/or graphics is possible [9].

Yu et al. [6] developed an assistive tool that includes a multimodal interface and a content aware plug-in. The solution uses a low-cost force feedback mouse to navigate the web and explore a web page layout via active haptic interaction. The cursor position of the mouse is constantly monitored and appropriate haptic and auditory
prompts are presented depending on the context of the user’s task. The prompts consist of guidance to the intended target and notifying users about elements present in close proximity to the cursor such as images and links. Evaluation of the interface revealed that haptic technologies played an influential role in assisting users to obtain a mental map of the layout of a web page. This is currently not possible with the use of screen readers. However, the study also revealed that the Haptic solution was comparatively slower than screen readers when searching and locating objects on a web page. Although the results of the study did help users gain a better understanding of the web page layout, some of the haptic mappings were ineffective; such as force feedback provided when the cursor was positioned over hyperlinks and in this case, users were repelled away from the link via the force reflection (which is counter-productive). Further, from the limited haptic feedback users had difficulty visualizing cursor position and speed. Auditory icons that provide an auditory description of the element, for example, a button, and earcons which provide simple sound feedback, for example, a click-sound upon button selection, both could have offered a greater level of description to the user when providing information concerning the location of the cursor. The interface mentioned in this article does not account for scrolling through web pages. The users in the study also found it difficult to work with the Logitech Wingman Force Feedback Mouse and further, they found the workspace to be too small and the haptic cues to be weak. The Wingman Mouse has 2 DoF and is restricted by the limited forces it can produce. A small workspace makes virtual objects seem smaller and would therefore constrain users from navigating freely. This study also indicates that a mouse may not be the best interface modality for a visually impaired user as they require a better grip and a larger workspace so that objects representation is larger and the elements are easier to recognize.

Kuber et al [8] designed a structured method to relay haptic sensations to the user by mapping force feedback to HTML elements. Findings from studies conducted with the users indicated that a structured design approach produced suitable haptic feedback where carefully tailored sensations allowed visually impaired users to differentiate between the mappings present on a page, and to locate and select targets quicker than those developed previously by Yu et al [6]. This paper discusses the use of a participatory design approach to develop haptic feedback for HTML elements. The user tests conducted were very useful in developing effective, meaningful force feedback, indicating success for a participatory design approach.

Kaklanis et al [2] developed a 3D Haptic web browser that combines visual and haptic modalities. The application represents a web page in a 3D environment that is enriched with haptic and aural feedback allowing visually impaired users to interact with a 3D scene via touch and interpretation from hearing. For each HTML element present in a web page, a corresponding 3D object, “hapget”, was generated. User participatory tests were conducted and revealed a major shortcoming in the work: users easily lost their sense of direction in the 3D space as they could not keep track of the cursor position while navigating the page.

Comai and Mazza [10] proposed a model-driven approach to demonstrate how haptic forces and interaction on the Web can be specified. The paper analyzes different characteristics of haptic devices and the typical, expected force reflection that can be provided during web page interactions. Some possible scenarios of enhancing web surfing with haptic interactions and appropriate haptic responses for HTML elements are discussed. For instance, a button may operate as a button in reality wherein the user is required to apply force in order for it to be pressed and as the user approaches the button on the web page a multidimensional spring effect could be added and applied to draw him/her to the center of the button. The authors developed a haptic-based web application as proof of their proposed concept in which a 3 DoF haptic device was used as a mouse to simulate movement on a flat 2D web page. User participatory tests were conducted to obtain feedback on web pages enhanced by haptics. It was determined that most users appreciated the haptic feedback and the assistive navigation that pulled them to the main content of the page and highlighted other important aspects of the web page. This work demonstrates a novel way to specify haptic interactions for 2D models. In the study however, the haptic interactions are designed for sighted users and although the haptic feedback proved to be effective for these users in enhancing their web browsing experience, it might not be so for visually impaired users. Previous research [8] has shown that visually impaired users find 3D interactive environments as a more effective way to navigate since the haptic feedback is richer and 3D objects make it easier for them to recognize the object and ‘feel’ their way around the environment.

3. PROPOSED AUDIO-HAPTIC SYSTEM

a. Overview

The suggested solution is an audio-haptic tool that combines web technologies with audio and Haptic interaction to assist the visually impaired with their perception of the spatial layout of a webpage, navigation of a webpage, access to graphical content and its interpretation. The solution functions as a web browser and uses an open source library to obtain the source code of a webpage, as opposed to a plug-in approach. A client-side URL transfer library is utilized to retrieve and load the source code of the web page and the file is transferred to an open source HTML parser to extract only the defined, valid HTML tags. A methodology has been developed to map the web page onto a 3D VR environment wherein the 3D space contains objects that represent the HTML elements and contain Haptic
information and associated algorithms. Audio is used to read the contents of the webpage aloud to the user using text-to-speech tools. For example, when a user moves to an image, the alternate text describing the image is read out. Text is articulated using synthetic text-to-speech voices that use International Phonetic Alphabets, which is the standard for most speech SDKs. The user is informed of the status of the webpage by announcing ‘page loading’, ‘page cannot load’, ‘page does not exist’, ‘page loaded’ and ‘no internet connection’. The software identifies the presence and location of HTML elements and determines landmarks such as headers, footers, sections, images, hyperlinks, image hyperlinks, buttons and radio/check boxes. The software then relays this information back to the user using audio and Haptic cues, depending on their position in the 3D application.

The main objective of the Haptic feedback is to inform users about the presence and position of web elements on the web page. Appropriate Haptic cues have been investigated throughout the design phase of this work. Assistive Haptic models that enable forces to be relayed back to the user through the Haptic device were designed and implemented. Further, when navigating a web page visually impaired users are informed as to their position, facilitating spatial awareness, and the content on a page, as well as assisting the user during their navigation to the desired section of the webpage. This functionality is linked to the toolbar application. Users may also scroll through web pages and are assisted in their orientation (position) on the page whilst scrolling. Appropriate Haptic sensations are mapped to HTML elements and may be accessed by the user on demand. These elements include images, image-hyperlinks, hyperlinks, page borders, search box and buttons, checkboxes and textboxes (such as Forms). The software monitors the user’s device movements and notifies him/her of the element touched and elements in close proximity. This facilitates spatial awareness for navigation and content retrieval.

b. Tools

Software implementation was performed in C++ within the Microsoft Visual Studio 2010 integrated development environment and with the packages Ogre3D, Ogre HAPI, Libcurl, C Lite Html and the Microsoft Speech SDK. Ogre3D is an open source 3D visualization engine that is scene oriented and written in C++, which encourages a larger developer base and for more rapid development of hardware accelerated 3D graphics applications. Many commercial products are developed using Ogre such as Ankh, Torchlight, and Garshasp which demonstrates its applicability as a stable, practical engine. Ogre has an active community that connects developers from all over the world. Feedback is received quickly and also the community sectors are divided into demographics for rapid connectivity. It does not support audio or physics modeling and as such, developers have the complete freedom to choose his/her preferable audio and physics engine to integrate with Ogre3D. The Ogre3D rendering engine is selected to create the VR environment that constitutes the 3D web browser for this research, in consideration of the aforementioned attributes. Ogre HAPI is selected as a compatible Haptic renderer to Ogre3D in this work. Ogre HAPI is a library that allows the developer to link VR engines to a device-independent Haptic API.

Libcurl is a library that is capable of transferring any kind of URL syntax such as FTP, FTPS, Gopher, HTTPS and SCP, and is chosen in this work to retrieve the source code from each web page. Libcurl is compatible with many operating systems; it builds and works equally across platforms. It is a client-side URL transfer library, it is portable, fast, thread-safe and IPv6 compatible. Libcurl can be binded with more than 40 languages. CLiteHtml is a HTML reader C++ class library that is an event based parser which raises events as it encounters various elements in the HTML document. Functions of CLiteHtml are utilized in the work to parse the HTML document and generate an event each time a defined tag is located. It proved to be simple, fast and efficient. Ogre 3D, Ogre HAPI, Libcurl and CLiteHTML are all open source solutions; free of cost.

Talib et al [11] recommend the use of Microsoft’s Speech Software Development Kit (SDK) for development of speech engines and version 5.1 is utilized in this work for text-to-speech implementation. The SDK contains Microsoft Win32-compatible speech application programming interface (SAPI), which has the capability to build complex applications that have vast voice functions. The SAPI supports speech recognition that converts user spoken words into text; adapting to the user’s speech patterns as he/she continues to use the system. The speech recognizer converts digitized sound waves into basic units of language sounds or phonemes from which words are constructed, analyzed and classified to their exact form, such as differentiating spoken words ‘write’ or ‘right’, and corrects spelling automatically. Two modes are supported which are dictation mode where the user can dictate information such as letters and reports, and command mode for the system to perform some task. It provides conversion of text to speech where the words are broken down into phonemes, analyzed to check speech that requires conversion such as to punctuation in a procedure called text normalization, and then displays the audio output. The audio output formatting is performed using linguistic rules and models which use the concatenative approach that is the Text to Speech (TTS) engine which concatenates real speech to natural (human) sound. User-friendly commands, ‘natural text commands’, can be created, avoiding a long list of commands. Its API supports languages such as C, C++, VB and script.

A low-cost Haptic device, the Novint Falcon (Novint), was selected for use in this work based on the
following criteria: DoF, workspace area, force capabilities in terms of maximum and minimum force output delivered by the device, size of the device, SDK and language supported by the hardware, and cost of the device. The Logitech Wingman Force Feedback Mouse has been discontinued and as such was not selected. The Phatom Omni (SensAble Technologies) offered only 3.3N output compared with 10N output from the Novint Falcon, but did offer a 6DoF compared with 3DoF of the Falcon. The price for the Omni (2400 USD) however was significantly higher than the Falcon (260 USD). The Falcon was selected primarily due to suitability of 3DoF for the application (torsional forces not necessary), force capabilities, SDK provided with relevant examples and online support, C++ development language, and low cost of the device. The comparison is tabulated below.

Table 1: Comparison of Commercial Haptic Devices Suitable for the Application

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Novint Falcon</th>
<th>Phantom Omni</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoF</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Workspace</td>
<td>4” x 4” x 4”</td>
<td>6.4” x 4.8” x 2.8”</td>
</tr>
<tr>
<td>Force Capabilities</td>
<td>2lb (c. 10N)</td>
<td>0.75lb (c. 3.3N)</td>
</tr>
<tr>
<td>Size</td>
<td>9” x 9” x 9”</td>
<td>5/6” x 8” x 8”</td>
</tr>
<tr>
<td>Extras</td>
<td>replaceable grips</td>
<td>Back drive function</td>
</tr>
<tr>
<td>SDK and language support</td>
<td>SDK provided, with relevant examples, native language C++</td>
<td>Open Haptics Toolkit (bundle containing multiple API’s, utilities, and source codes)</td>
</tr>
<tr>
<td>Cost</td>
<td>260 USD</td>
<td>2400 USD</td>
</tr>
</tbody>
</table>

**c. System Design**

An open source library, libcurl, is used to extract HTML content from the Web. The HTML code is transferred to a HTML parser, to parse and extract all valid tags defined within the system. The list of these recognized tags progresses through a series of operations to include speech synthesis and Haptic force feedback to the HTML Web elements. The System Architecture is shown in Figure 1, with Figure 2 representing a conceptual view of the system. Figure 3 represents a Swim Lane Diagram, which visualizes connectivity between design components and fundamental aspects of implementation. The web browser is capable of rendering graphical content (3D representations) in a VR environment. The system allows the user to navigate the web page with the help of a Haptic device. The browser helps the user to recognize the position of different web elements and provides force effects as the user explores each element. Audio feedback is provided with each 3D model, prompted as the cursor is positioned over the web element.

As the user enters a URL, the libcurl reads and loads the web page internally, and retrieves the source code of the page. The source code is transferred to cLiteHTML and verification is performed to check if the document exists. If no document is found, the user is informed via an error message. For a correct URL, the document is parsed and those HTML tags that have been defined are extracted, and pushed onto the stack. The GUI elements are loaded and iteration takes place for the stack simultaneously, for respective GUI objects that are mapped, for each element. After the iteration and mapping is completed, each element is added to the list of Ogre components. The visualization of Web elements begins in Ogre; here, all of the objects, their positions, their force models, and audio feedback are attached to a Node within the Scene (SceneNode), representing entity creation. After all entities are attached to a Scene Node, these are attached to a Scene Manager, with ambient light added for the scene and finally, scene rendering is invoked and hence the scene is visualized for the user. As the user explores the Web page and finds the desired link, which may be in the form of a button, hyperlink or hypermedia (image etc), once the link is activated; the URL is subject to change (as a new Web page is invoked). Hence the entire process recommences. As the user finishes his/her task on the Web, the user will close down the browser, and the processing ceases until the user again activates the browser. The primary operations that the user can perform includes navigation to a website, exploration of...
a web page, conducting a search within the web page and activation of a link. At the back end, the system is responsible for providing the force effect, reading web element content, reading web content and loading a new website or web page.

![Swim Lane Diagram](image)

**Fig 3: Swim Lane Diagram**

### d. Implementation

Within Ogre3D, implementation was coded in C++, exploiting classes and their inheritance. The main classes and methods are described below.

### e. Haptic Browser Class

This class handles all the tasks, starting from reading the URL to creating the entire Ogre browser scene displaying the hapget. It is responsible for delegating tasks to the various classes and accomplishes the tasks of taking the URL input from the user, parsing the HTML page and acquiring the list of objects to be displayed. This class is also responsible for creating a scene and rendering the objects corresponding to the various tags on the HTML page to the Ogre browser. Methods within the Haptic Browser class include main, create Scene, create Entity, create Frame Listener and create Browser. The main() method handles the execution of all project components and displays the hapget to the scene. It attaches the hapget to the Ogre scene manager to be displayed and processed further. It binds the objects to be displayed on the scene and internally handles the work of rendering all the hapgets on the scene.

For each object in the list of displayable elements, an entity is required to map the actual object that is displayed on the normal browser and the corresponding mesh component that needs to be loaded on the Ogre browser; the create Entity() method is responsible for this action. This method handles the creation of the Ogre object corresponding to each of the entities observed on the normal browser and returns it to the main create Scene() which calls this method. The create Frame Listener() method applies a Frame listener to the Ogre scene to be able to get the Haptic device input from the user. The frame listener captures any actions performed by the Haptic device connected to the machine and is able to reciprocate to those actions by invoking the right event handler for the action. For example, the frame listener is responsible for taking the input of the device hovering on a hapget and read out the associated name/content of a hapget aloud to the user. The Haptic Browser() method is the default constructor of the class and initializes the Haptic device attached to the machine. It also adds a frame listener to the active Ogre frame.

### f. Html Page Reader Class

This class reads the entire HTML page source code from the URL. It initializes and reads the source using Curl. The method ensures that the input document is a proper HTML document and reads the HTML tags using a custom HTML Parser. Methods include load Page and getHtmlSource. Load Page() loads the HTML page based on the URL entered. The Curl components are initialized here and the page contents loaded using Curl. The method get Html Source() returns data read from the page source, as a string.

### g. Html GUI Object Class

The purpose of this class is to define a GUI object to be shown on the Haptic browser. If this object is referable, then the class will define the Hapget to be rendered. The name and contents within the base description of the GUI object that is read from the HTML are defined.

### h. Html Parser Event Handler Class

This is the main parsing event handler class. The class defines the various callback functions that are invoked by the Html Parser for the occurrence of various events
during the parsing procedure. Data members include the html Obj List which holds a list of HTML tags and all of the properties associated with the tags. Member functions include Begin Parse, SmartTag and End Tag. Begin Parse() internally indicates the start of the parsing procedure. SmartTag() is invoked when the parser comes across a HTML open tag. This method validates the tag that is opened, creates a Html GUI Object by reading all the properties associated with the HTML tag, adding it to the list of tags using a custom stag and the tag is pushed onto the custom stack to indicate that the tag was opened, and not yet closed. The End Tag() method is invoked when a HTML tag is closed. The items are popped from the custom stack until the closing tag is received. The items popped from the stack are pushed into the final object list that will contain all identified and recognized tags.

i. Haptic Browser Frame Listener Class

The frame listener handles the integration of the HAPI integration with the Ogre framework. It allows adding a listener to the Ogre scene to capture all the triggered events and calls the corresponding action events for the action performed by the user. Methods include the Haptic Browser Frame Listener(), which initializes and sets the camera offset, as well as setting the scale of the scene in Ogre, and process Custom Key Input() which handles the key events using a custom implemented frame listener. All the events to be triggered on specific key strokes or the input from the Haptic device are capture in this latter method and process further. The speech integration upon device hover over the hapgets is registered in this class and related action is performed.

ej. Graphical User Interface

To represent each HTML element, a corresponding 3D shape was created. These 3D shapes are rendered in a scene as the whole rendering process begins and are distinct from each other. The goal behind the latter is to create shapes that would be easily recognized by visually impaired web users. Figure 4 provides some sample shapes created using Autodesk 3Ds Max 9.

The models were exported as three file types, .mesh, .material and an image file, to create a compatible Ogre format. The general format is shown in an example provided in Figure 5.

```plaintext
//This is a comment
Material walls/funkywall1
{technique //first, preferred technique
 {pass //first pass
  {ambient 0.5 0.5 0.5
   diffuse 1.0 1.0 1.0
   texture_unit //Texture unit 0
    {texture wibbly.jpg
     scroll_anim 0.1 0.0
     wave_xform scale sine 0.0 0.7 0.0 1.0}
   //Texture unit 1 (multitexture pass)
   texture_unit
    {texture wobbly.png
     rotate_anim 0.25
     colour_op add})
  //Second technique; uses a fallback or
  //LOD level
  technique
   { // ...and so on}}
```

Fig 5: Example of code within a .material file.

The .mesh file is a binary file in Ogre format and is specially-made to work with the Ogre application. The 3D model meshes are used by DirectX application and gaming. They are often used for rendering game objects and other entities in 3D games. This file creates a digital representation of geometric models by using a collection of polygons to form the surface of each object. The .material file type defines several material types within the one sole script. The format of the script is pseudo C++, with sections delimited by ‘{’, ‘}’, and comments indicated by starting a line with ‘//’ (note that no nested form comments are allowed).

Images represented as formats .jpg, .png, .bmp, are associated with the .mesh and .material files; the picture can be of such format and is used as a texture for the model, contained in the scene as it is rendered. Figure 6 shows a screen shot of the GUI, with the hapget symbols provided.

---

Fig 4: Samples of some HTML elements and corresponding 3D shapes.

Fig 5: Example of code within a .material file.

Fig 6: Screen shot of the GUI, containing a textured background with hapget representations.
4. RESULTS AND DISCUSSION

The classes described previously were implemented and an overview of class interactivity is summarized in this section; thereafter the results of testing these class implementations are provided. The application utilizes the 3D Ogre graphical rendering engine and there are many classes generated within Ogre to render the scene.

The main framework consists of two classes: Base Application and Haptic Browser. Base Application() is a virtual class since it contains a virtual function, create Scene. It cannot be instantiated on its own and is meant to be used as a base class. Haptic Browser is derived from Base Application and thus needs to implement the create Scene function and other inherited functions. The Base Application consists of five listeners: SDK Tray Listener, Key Listener, Mouse Listener, Frame Listener and Window Event Listener. Within the Frame Listener, there are notifications to be received before and after a frame (model) has been rendered to the scene. The SDK Tray Listener is used for GUI widgets and camera handling. The Key Listener is used to handle different keyboard events and the Mouse Listener is used to handle different mouse events. The Window Event Listener is used to handle window-related events, such as resizing or moving the current window. Figure 7 shows an overview of the main framework.

The Haptic Browser is the main class of the application. It contains the code that constitutes the browser (including the Scene, Cameras, Lights, Physics, Audio, and GUI). The main function is the Base Application::create

Scene, and consists of all the elements to be rendered on the screen. The Haptic Browser class consists of the function Haptic Browser that has the main workflow of the project. The workflow involves obtaining the web page, extracting defined HTML tags and their attributes, and converting them from html GUI Objects to hapgets, and positioning them in the scene. The constructor initializes all the fields required for haptic rendering. M Haptics Device holds a pointer to a HAPI::Any Haptics Device, which is any haptics device supported by the HAPI package. mSpace holds a pointer to the haptic space where all haptic objects will have to be added to do haptic rendering. The destructor is used to delete the haptic world. Furthermore, there are other functions like Create Frame Listener that is used to add the frame listener for Ogre HAPI and to add it to the root. It will keep updating the haptic rendering with the specified Scene Manager projecting the rendering so that it is relative to the camera position and orientation. The object created here is mHAPI Listener.

Ogre fundamentals include the Scene Manager, Entities and the Scene Node. The Scene Manager keeps track of location of all the objects visible on the screen such as lights, camera, entity and planes. Entities are all those objects that can be rendered on the screen. The 3D model of hyperlink is an entity. However, lights and cameras are not entities. To add an entity to a scene, the entity must be attached to a Scene Node object. Every object in the scene must be attached to a Scene Node because it contains information about the location and orientation of these objects. Any number of nodes can be attached to the Scene Node. In the function Create Entity, one Scene Node is created which can be called multiple times to create multiple scenes. An entity is attached to a node and positions are set for the node. An object of Ogre HAPI is created and force feedback model is applied to the respective loaded object. The object of Ogre HAPI is being attached to the Scene Node and finally, the object is added to the scene. In the function Create Scene, an Ogre scene is created by loading the entire components specific to the scene. If there is any node still used by the Scene Manager it will be removed. There is a loop through all the Ogre GUI Object (containing the attributes of a mesh file) and these objects are attached to a scene node along with their positions. This is followed by the setting of ambient light within the scene, and position in the scene. As mentioned previously, Haptic Browser is the main function of the application. Its function is to obtain the page, parse the document and map the extracted web elements into appropriate 3D models along with their positions. Finally, the models are being added to the list of Ogre components which will be rendered on the Scene using the function create Scene.

Fig 7: Overview of Ogre Framework with constituent classes.

a. Main Workflow
Components of the main workflow include extraction of the HTML source code, parsing the HTML content, inclusion of force feedback and audio feedback, and scene rendering. To extract the HTML source code, the application uses Curl; this enables retrieval of the HTML content from a given URL. The HTML source code is saved in a string and passed to the HTML parser. The parser is required to extract the valid HTML tags. HTML content is provided to the parser using the function set Html Document (html Content).

The parser will begin parsing the document, and utilizes the class HTML Parser Event Handler for different event handling tasks. Events are raised when the tags found, such as Start Tag (), to validate the data, push it onto the stack and End Tag() to pop the items from the stack, until the close tag is received. Whenever the parser comes across a piece of text, the characters () event is invoked, and the parser saves the text separately. After each tag has been parsed, the stack pops and adds the tag to a list containing all identified validated tags.

For all the valid tags defined in the list, appropriate hapgets (3D mesh files) defined in the list are loaded. Iteration takes place through the list to find the tag type and then the appropriate hapget will be mapped to the HTML Web element it represents. Speech is also mapped to the Web elements. The position of each Web element is loaded and set for the hapgets, within the 3D environment. As the mapping is progressing, the hapgets are added to the list of Ogre components.

Scene rendering enables 3D display and captures topographical changes of the virtual environment, as initiated by the user or via time-driven events. This part of the implementation is divided into two functions: create Scene() and create Entity(). In the create Scene() function, the entities are added to the Scene Node and the entities comprise all the mesh files (Ogre Gui Objects). Their positions and scales are also attached to the Scene Node. In the create Entity() function, the iteration is executed through the list of Ogre components and they are attached to a Scene Node. Lighting properties are also added to the Ogre scene. The Scene Node is then attached to the overarching Scene Manager.

b. Testing

In the current version of the software, four HTML elements are preliminarily rendered within the scene and these include: button, text, hyperlink and image content. The user may interact with each element to receive aural and/or Haptic feedback, and navigate the Web. Five major test cases were effectively deployed, which included Web browsing, exploration of the Web, activation of a link, exploration of an image and reading Web content. Stand-alone testing was further performed on image-based content with various Haptic representations explored, including edge detection and three holding methods.

The first test case involves the user initiating Web browsing, with the browser initially being inactive (not running). A URL is provided to the system and the website is effectively loaded with appropriate 3D objects (hapgets) placed within the scene. The next test case verified user exploration of a website. A loaded website was made available on the screen and the user effectively moved through the hyperlinks and hypermedia (including image-based content). The user hears the word ‘image’ upon placement over the image content. In a separate test case, for greater exploration of an image, as the mouse is moved across the image, an aural description of the image is provided; for example ‘Google logo’, as was one scenario for this test case.

In another test case, simple force information is provided upon exploration of the image data, including force effects involving magnetic, frictional (starting, dynamic and stopping friction set) and spring-damper motion. The simple force effects are to be replaced with more descriptive Haptic representations of the image data in future implementations, by use of edge detection, filtering and threshold-based approaches. Although these were effectively implemented in a separate application (via MATLAB, Math works Inc.), the inclusion of these greater fidelity models were not implemented within Ogre due to further optimization requirements for minimization of datasets, for real-time user interactions.

A test case was effectively executed for activation of a hyperlink, where a loaded website was made available on the screen and the user moved through the hyperlink ‘Gmail’; in one scenario. The user clicked as soon as the audio feedback was provided and the output was that the system loaded a new Web page for the user of the Gmail account. In the final test case for reading Web content, a loaded website was made available to the user on the screen and as the user explored the Web page, which was Google for one scenario, and moves the mouse cursor passively over the text, the system read aloud the respective text, providing the aural feedback. In the scenario given, the system read out ‘Google.ae offered in Arabic’.

5. CONCLUSIONS AND FUTURE WORK

An application has been developed to assist visually impaired citizens with greater access and navigation of the World Wide Web. The system provides audio and Haptic feedback as users interactively navigate the Web and these cues facilitate an enhanced interpretation of website content for those with impaired sight. HTML elements, including buttons, text, hypermedia and images are mapped to 3D objects within a VR environment, with hapgets that have
force representations including magnetism and friction. Audio feedback converts text into speech, as the user passively moves the mouse of the related text. Testing reveals that main functionalities are implemented effectively in real-time, including Web navigation (moving between Web pages) using hyperlinks and hypermedia, audio cues are provided as associated with the element (text and images), and preliminary haptic feedback is provided, with simple force effects. The system is implemented in open source and using a low cost Haptic device, the NovInt Falcon, which makes the application accessible for both developers and users.

Future work will focus on integration of more complex force models of image-based content and other multimedia content, such as using edge detection, thresholding, filtering and/or bump mapping, to provide a more detailed Haptic representation of the image or media explored by the user. Further, user-tests must be conducted to determine the degree of acceptability for partially sighted and/or blind users; and customization enabled for a more personalized experience, depending on user preferences. Speech input for user-driven voice commands, for navigation between Web pages is also a consideration for future implementations and currently the authors are considering Microsoft Speech API to introduce this feature. Support for other Web technologies, including browser Favorites, History and Help, as well as interactive selection of 3D representations of Web elements, are considerations for future versions of the application.

REFERENCES


