Haptic Applications Meta-Language

Fayez R. El-Far, Mohamad Eid, Mauricio Orozco, and Abdulmotaleb El Saddik
Multimedia Communications Research Laboratory - MCRLab
School of Information Technology and Engineering - University of Ottawa
Ottawa, Ontario, K1N 6N5, Canada
{fayez, eid, morozco, abed} @ mcrlab.uottawa.ca

Abstract

A wide range of haptic devices exist that possess the potential to offer users a rich experience in a virtual reality environment. This however depends on the haptic device to be used. Surely, removing the burden of users having to 'adjust to' operating haptic devices is welcomed. In this paper, we propose the creation of the Haptic Applications Meta Language – hereon HAML – which is an XML-based language created to describe Haptic-enabled frameworks to a high degree of detail. The envisioned goal of HAML is to allow for the creation of plug-and-play environments in which a wide array of supported haptic devices can be used in a multitude of virtual environments, with the compatibility issues being handled by automated engines instead of programmatically by the user. Therefore, we introduce the HAML framework and discuss its tentative structure, proof-of-concept implementation, and avenues for future work.

1. Introduction

Haptics, a term which was derived from the Greek verb "haptesthai" meaning "to touch", refers to the science of sensing and manipulation through touch [1]. Since the emergence of PHANToM series haptic interfaces from SensAble Inc. in 1990s [2], haptic devices are getting more diversified and cheaper in the market now. For instance, a desktop haptic device, called Novint Falcon from Novint Technoligies Inc., is going to be released in the coming spring, which will cost only about $100 [3]. Therefore, we anticipate that a haptic device will be deployed with PCs for personal customers in the near future. Nowadays we are able to download audio or video clips, and play them back with some standard commercial players, such as RealOne Player or Windows Media Player. Since these videos are coded following certain standard formats, they are highly independent from the graphics cards or displays we are using. Similarly, imagine someday, people are able to sit in front of a haptic equipped PC, download some haptic-enabled antique clips from the online museums, and play them like we are holding the real pieces regardless of the device that has been used. It sounds exciting indeed. But are we ready for this new medium goes through the internet and enters our lives like audio or video does? We believe that, before that happens, a technology-neutral format to describe the haptic model and to solve the incompatibilities between different haptic devices is definitely needed.

Furthermore, the process of incorporating support for multiple haptic devices in a single virtual environment is non-trivial. This is because each haptic device requires a development API that includes specific haptic rendering algorithms, compatible collision detection algorithms, and – in some cases – limited support for general graphical tools. In this paper, we develop a novel XML-based meta-language that is comprehensive, scalable, and extensible enough to provide the models and XML schema encodings for the representation of a generic haptic application including application general information, haptic device and its characteristics, the haptic rendering, visual rendering, and the system behavior, etc. The remainder of the paper is organized as follows: in section 2, we present an overview of related work, the proposed HAML framework, and the tentative representation model of HAML schema. Section 3 describes implementation related concepts and preliminary results. Finally, in section 4 we pinpoint related issues and our immediate future work.

2. HAML Meta-Language

2.1. Related Work

The idea of developing a meta-language for haptic applications is, as far as we know, entirely new. As a matter of fact, existing modeling languages such as SensorML [4] or VRML and X3D can describe – to a certain extent – the haptic device and the virtual environment respectively. However, as for SensorML, considerable efforts need to be spent in order to describe a haptic application because it merely describes the haptic interface as a sensor and provides no means for modeling the virtual environment aspects of the application. On the other hand, VRML is a text-based meta-data based modeling language to describe virtual, 3D, interactive environments [5]. Another 3D Scene description language which has superseded VRML is Web3D Consortium's X3D [6]. VRML and X3D fall short

Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'06)
0-7695-2697-7/06 $20.00 © 2006 IEEE

in describing the haptic interface hardware and consequently, tailoring the virtual environment to fit a particular haptic device is tiresome, low-level, and programmatic endeavor. The goal of HAML is to intuitively solve this problem by providing a highly descriptive document that enables an interpretive backend engine to discern and solve compatibility issues.

The work in [7] proposed a novel XML-based approach to represent generic haptic application. The model includes: application general information, haptic interface, haptic and visual rendering, and the system behavior, among others. However, many key features related to tactile and haptic interaction is not yet covered. Meanwhile, an ongoing effort to introduce a work plan for the development of a new set of ISO standards for tactile and haptic interactions, based on the GOTHI model [8], are elaborated in [9]. These standards will provide ergonomic requirements and recommendations for haptic and tactile hardware and software, and guidance related to the design and evaluation of hardware, software, and combinations of hardware and software interactions. Even though the proposed draft for the standard provides comprehensive guidelines for haptic interactions, it lacks the description of the haptic application and the virtual environment component. On the other hand, HAML describes all aspects related to a haptic application. Therefore, HAML is a meta-language that is literally comprehensive, scalable, and extensible.

2.2. HAML Framework

The HAML framework is designed to prove that the HAML description could be utilized to make devices, APIs, and their corresponding rendering/collision algorithms almost irrelevant. In order to accomplish this goal, the haptic component is separated from the virtual environment so that the haptic API could be changed without affecting the environment. The basic components of the HAML framework are shown in Figure 1.

The user interacts with the whole framework via the GUI component that captures the basic user requirements (such as the interaction type/device, the virtual environment components, data recording, etc.). These requirements are then passed through the translation engine, which relies on the HAML schema to "pump-out" a HAML-formatted document. This document holds a startup/default configuration of the haptic application required for the framework to work - a discussion of the structuring of HAML will be held later in section 2.3. The Authoring Agent (AA) parses the HAML file to dynamically create the haptic application by selecting and composing components – haptic device, rendering engines, collision detection engines, graphic components, and APIs – that meet the specifications defined in the HAML file and are compatible as well. Notice that the HAML repository stores HAML-formatted description for all available devices, haptic and graphic APIs, and all related information. At this stage, the HAML document is not yet complete, as we are missing the information regarding the newly authored virtual environment application. After the environment has been created and finalized, a "commit" function is performed to update the HAML document accordingly, which means that the HAML document is no longer static.

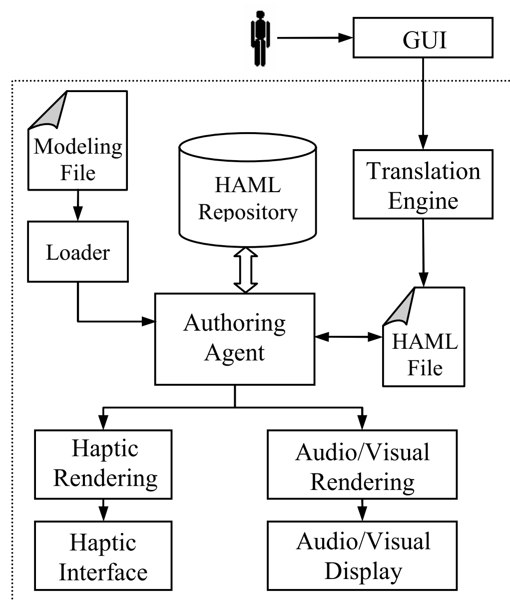

Figure 1. HAML framework.

2.3. HAML Structure Overview

As mentioned earlier, HAML is a haptic-related information, XML based schema meant to describe the haptic device, API, rendering engines (haptic/graphic), as well as any other pieces of information pertaining to the description of the haptic framework such as the device vendor information, framework hardware list (PC processor, memory capacity, etc.), the HAML document creation/modification dates, any intellectual properties, any special considerations, etc.

As it stands, the structure of HAML consists of six (6) main categories. It is important to note that this is not the final structure of HAML, and is open to modification as new needs arise. A short example of a HAML document is presented in Figure 2. These six categories are as follows:

1. Haptic Device: This category attempts to fully describe the haptic device itself. Information about the haptic device include, but are not exclusive to:
   a. Device Identification: Manufacturer, Serial Number, Model Number, etc.

Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'06)
0-7695-2697-7/06 $20.00 © 2006 IEEE

b. Device Characteristics: Control type, degrees of freedom, position resolution, force sensitivity.
c. Compatibility information: Haptic development API, Graphical Development API, Operating System, etc.

```
<!--------HAML Excerpt------->
<?xml version = "1.0" encoding = "UTF-8"?>
  <HAML xmlns =
http://www.mcrlab.uottawa.ca/HAML>
<Haptic_Device>
  <Device_ID>
      <Model_No>1234XYZ</Model_No>
      <Serial_No>1234567</Serial_No>
  </Device_ID>
  <Device_Characteristics>
      <DOF>
         <Count>6</Count>
      </DOF>
      <Joint_Angles>
         <Resolution>
            <Quantity>0.01</Quantity>
         </Resolution>
      </JointAngles>
  </Device_Characteristics>
</Haptic_Device>
<Haptic_API>
  <General>
      <Name>OpenHaptics</Name>
      <Version_Mjr>1</Version_Mjr>
      <Version_Mnr>0</Version_Mnr>
  </General>
</Haptic_API>
</HAML>
<!--------End Excerpt----------->
```

Figure 2. An Excerpt of a HAML Document.

2. Flags and Data Types: As mentioned earlier, HAML will be a process modeling language. Each process receives an input and delivers an output. These I/O operations require a definition of data types and error/configuration flags in order to continue the process flow. The category includes:
   a. Data Types
      i. Simple Data Types: Such as floating point *Quantities*, integer *Counts*, *Time* Values, *Boolean* flags.
      ii. Aggregate Data Types: Such as *Arrays* of simple data types, *ranges* of simple data types, etc.
   b. Error and Configuration Flags: Each data type has a set of possible characteristics that define and restrict its usage and implementation, for example the *Quantity* data type could have the following properties: *Unit Type* (floating point), *Unit of Measure* (cm's, m's, kg's, etc), *Min/Max* (ordered pair of the minimum allowable value, and the maximum allowable value), *Fixed/Dynamic* (Boolean flag that dictates whether this variable can be modified dynamically.

3. Haptic Development API: This category describes the haptic development API and includes details of:
   a. API General Information: Such as programming language, release version, API support information.
   b. Haptic Rendering Engine: Implementation details of the haptic rendering engine and the collision detection algorithms.
   c. Haptic Performance: Characteristics about the haptic performance such as haptic resolution, force depth, impedance ratio, force update rate.

4. Graphics Development API: All general and programmatic (non-scene describing) graphics-related information are grouped in this category:
   a. Graphic Rendering API General Information: Such as language, release version, API support information.
   b. Display Device Requirements: Such as required resolution, depth buffer bits, display frequency.
   c. Preprocessing requirements for haptic rendering.

5. Virtual Environment: Typically, a VRML flavor, or X3D is used to describe the virtual scene, however, such languages might not be enough to describe the haptic characteristics of the virtual world, and thus might need extension or even us creating a new description language. The description of the virtual scene must include haptic characteristics of the objects in the environment in order to make the virtual world 'haptically-similar' regardless of the device being used.

6. Meta Metadata: Information in this category are meant to describe the HAML document instance being created, such information includes:
   a. Author Information: Name, Location, Contact, Company, Purpose, Support Information, etc.
   b. Dates: Creation date, Modification Date(s), Release Date(s)
   c. Intellectual Property: Copyright notices, patents, usage restrictions, etc.
   d. Document Information: HAML Version, Document Location (network or local).
   e. Notes and Need-to-Know Information: Space reserved for author/reviewers/users to post information that has no other place.

3. Implementation

As a proof of concept we propose using two (2) different haptic-devices with different APIs that greatly differ from one another. Our choice for these devices was SensAble

OMNI and the Immersion CyberGrasp [10]; the SensAble OMNI is a single-point contact device whereas CyberGrasp is a multiple-point contact device. As per APIs, we looked at several APIs including OpenHaptics [11], GHOST SDK, CHAI 3D [12], H3D, and Virtual Hand Toolkit (VHT). Of the five APIs only VHT was designed for multiple contact haptic devices whereas the rest were compatible with single point contact haptic devices, and thus was chosen. CHAI 3D was later chosen over the other single point APIs, because it allows separating the haptics scene from the graphics scene. By choosing two very different APIs we could prove that multiple API switching is possible.

After fully describing the two haptic devices using HAML, a generic, well-described virtual environment is created. We developed an application, using the Sensable OMNI and CyberGrasp, comprising a cylinder and a plane and the user was asked to grasp the cylinder and collide it with the plane. Using either device with their corresponding API, the user was able to interact with the environment. Without any change to the virtual environment, we were able to change the haptic module to use CHAI 3D with the OMNI; thanks to HAML. The OMNI was also able to interact with the environment by probing the object or feeling the stiffness and texture of the cylinder and the plane. This demonstrates that by utilizing HAML descriptions, the backend engine is capable of handling multiple APIs and at the same time takes advantage of the capabilities of different commercial haptic devices. Figure 3 shows a snapshot of the same application using the OMNI and the CyberGrasp devices.

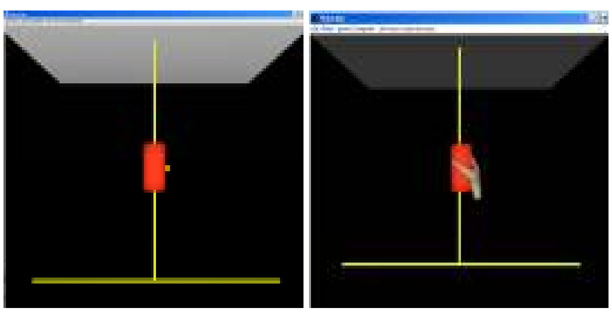

Figure 3. Snapshot of the application using OMNI (Single contact point with the VE) and CyberGrasp (multiple contact point with the VE)

4. Known issues and future work

As part of our research, we were faced with the question of necessity: Is it necessary to create a new meta-language for haptic environments or could these environments be sufficiently described using a mélange of existing meta-language, such as SensorML, VRML, or X3D. Even though this issue warrants further research, it seems that such a mixture would still yield an incomplete description of this environment as it does not sufficiently describe the software backend of the rendering engines (haptic and graphic) which is so essential to resolving compatibility issues between devices.

An integral part of our future work is finalizing the HAML schema, and extending it considerably. As mentioned earlier, the diversity of devices and their methodology of implementation should be catered to as much as possible. Another issue is the feasibility of such a project. Admittedly, it is an ambitious project; one that could change the face of haptic research fundamentally.

6. References

[1] J. K. Salisbury, and M. A. Srinivasan, "Sections on Haptics, In Virtual Environment Technology for Training (BBN Report No. 7661)", Cambridge, USA: The Virtual Environment and Teleoperator Research Consortium (VETREC) affiliated with MIT.

[2] SensAble Technologies: Products Page, OMNI Phantom (2006). Available online at: http://www.sensable.com/products/phantom_ghost/phantom-omni.asp.

[3] Novint Technologies, Inc., http://www.novint.com/.

[4] SensorML developer website (2006), Available online at http://vast.uah.edu/SensorML/.

[5] Web3D Consortium, "The Virtual Reality Modeling Language", accessed on 05/05/2006, http://www.w3.org/MarkUp/VRML/

[6] Web3D Consortium, "X3D", accessed on 05/31/2006, http://www.web3d.org/.

[7] Zhou J., Shen X., Shakra I., El Saddik A., and Georganas N. D. XML-based Representation of Haptic Information. In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2005.

[8] Carter,J., van Erp, J.,, Fourney,D., Fukuzumi, S., Gardner,J., Horiuchi, Y., Jansson, G., J_rgensen, H., Kadefors, R., Kobayashi, T., Kwok, M.G., Miyagi, M., Nesbitt, K.V. The GOTHI Model of Tactile and Haptic Interaction. In Proceedings of GOTHI'05 Guidelines on Tactile and Haptic Interactions, October 24-26, 2005.

[9] Van Erp, J. B.F., Andrew, I., and Carter, J. ISO's Work on Tactile and Haptic Interaction Guidelines. In proceedings of the EuroHaptics 2006, paris, France 2006.

[10] Immersion Technologies, 3D Interactions Product Page (2006). Available online at: http://www.immersion.com/3d/products/cyber_grasp.php

[11] SensAble Technologies, "Open Haptics Toolkit", accessed on 05/31/2006, http://www.sensable.com/products/phantom_ghost/OpenHapticsToolkit-intro.asp.

[12] CHAI 3D, "The Open Source Haptics Project", accessed on 05/28/2006, http://www.chai3d.org/.

0-7695-2697-7/06 $20.00 © 2006 IEEE