

The CHAI libraries

F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris
L. Sentis, E. Vileshin, J. Warren, O. Khatib, K. Salisbury

Computer Science Department, Stanford University, Stanford CA 94305, USA,
[conti, barbagli]@stanford.edu

Abstract. In this paper we present CHAI, a set of haptic/graphics C++ libraries that were developed in the last years by a varied group of researchers. The CHAI libraries allow both high level and low level programming of haptic applications. Users that are not interested in implementation details can easily build visual-haptic scenes using a large set of pre-implemented algorithms. However, differently from other existing haptic libraries, the CHAI libraries allow users to tweak with low level details, such as changing the control algorithms for supported devices or adding new custom devices. CHAI is a growing project, one featuring the efforts of many researchers around the world. A steering committee will periodically review any proposed addition to the libraries and publish new extended releases. Some demos created using CHAI, as well as the main features of the libraries, will be demonstrated during Eurohaptics demo session. In such occasions CD-ROMs containing the latest release of CHAI, as well as its documentation, will be freely handed out to anyone who is interested.

1 Introduction

The idea of creating a set of haptic/graphic libraries, allowing anyone who works with haptic devices to re-use a basic core of well-known algorithms, is an obvious one. Various haptic/graphic Software Development Kits (SDK) or Application Program Interfaces (API) have been developed in the past. Originally such libraries were developed for a specific device by the company that designed it. Such is the case of Microsoft's DirectInput API for the SideWinder force-feedback Joystick or Immersion's TouchSense API for the Wingman force-feedback mouse. From the haptic research community perspective, however, the best known of such libraries is the GHOST SDK created by SensAble technologies specifically for their Phantom device [1]. GHOST was the first of such libraries to be largely distributed. The commercial success of the Phantom in the haptic research community, as well as the lack of any easy to use alternative to communicate with it (no free low level API is given with the device) has made GHOST the first set of libraries used by many haptic programmers.

The problems with GHOST and other proprietary APIs from a researcher perspective are clear. All of such APIs only support one device type, they can be very expensive and they are not open source and thus not expandable. While

they may work for creating simple demos their usage in a research environment, where new devices and new algorithms are designed and tested everyday, is limited.

The Swedish based company ReachIn developed and commercialized the first haptic/graphic API that was independent from a specific haptic device[7]. The ReachIn API can be used in conjunction with Phantom[1] and Delta[2] haptic devices as well as their proprietary ReachIn display that allows for graphic and haptic co-location. While the ReachIn API allows for more freedom in usage with respect to device-specific APIs, it still does not have some of the advantages of open-source software.

Novint technologies, an R&D company that operates in the field of haptics, was the first to launch a set of open-module haptic/graphic libraries. Their *e-touch* API [6] was launched in 2001 with an open module license and was adopted by various institutions. The e-touch API supports both Phantom and Delta devices and allows users to create haptic scenes featuring rigid meshes and simpler implicit-surface haptic objects. Moreover it allows to create a haptic-based desktop environment where menus can be accessed using the haptic device. E-touch is an open module, and therefore can be expanded and modified.

In the following we will introduce the main features of the CHAI libraries and what sets it apart from past haptic/graphic APIs.

2 Main features of CHAI

The motivation behind the CHAI libraries was to create an API that would serve two different types of users: high level programmers that need an easy to use API for creating haptic demos and haptic researchers that need to be able to experiment with new devices and new algorithms while not wanting to re-write all the code from scratch. The following key elements were identified as of basic importance:

- ease of use and consequent appeal as a tool to teach introductory classes on haptics.
- to be open source and easily expandable in order to be used in a variety of different projects with very different scopes;
- to allow users to easily add new hardware, new haptic rendering algorithms and new simulation techniques for rigid and deformable objects;
- to be usable with different compilers and different C++ based programming environments;
- to be a free and growing tool in the hands of the haptic research community;

In order to address the points above the CHAI libraries have been designed with the following features:

- CHAI is C++ based and can be used from Microsoft Visual C++ and Borland C++ Builder. While CHAI is currently only Windows based extensions to Linux, RTAI-Linux, and Mac OS X are being considered.

- CHAI can be used as a high level tool to quickly create haptic scenes. The structure of a typical haptic scene is simple: the base class *world* can be populated by various virtual objects (based on implicit surfaces or on meshes) whose physical characteristics (mass, texture, color, initial position and velocity, etc) can be saved in an XML file. OpenGL-based graphic rendering of objects and tools is automatically taken care of by the CHAI engine. Basic haptic rendering algorithms, such as the god-object [5] and proxy [4] algorithms or the god object for implicit surfaces[8], are also implemented and can be used to compute the haptic rendering for a given virtual scene. Ideally we envision other algorithms being added through time by anyone interested in using them in a CHAI context. Sound rendering and texture rendering are examples of algorithms that we believe should soon be added to these libraries.
- CHAI supports all major 3 degrees of freedom commercially available haptic devices (Phantom and Delta devices). However CHAI allows users to easily control and write drivers for any additional device. The class *additional devices* creates a standard interface for haptic devices which can be used to create drivers for any new device. In order to make this simpler the class *boards* supports various control boards and creates a standard interface for more control boards used by different research groups (currently some boards by Servotogo [9] and Sensoray [10] are supported).
- CHAI allows users to define new interaction paradigms for a same device. Most current libraries only allow the classic single point contact interaction modality. CHAI, on the contrary, allows users to define their own interaction paradigm. A Phantom, for instance, can interact with virtual objects using a single point contact paradigm as well as multiple points, a line contact paradigm, a surface contact paradigm or any other type of interaction that a user may need. The key element to accomplish this is the distinction between the *tool* class and the lower level *device* class. A device is a lower level entity: users can read its generalized position (linear and angular) and send it a generalized force vector (linear forces, torques). A tool, on the other side, is the entity that interacts with the virtual environment. Users can easily connect a generic tool with any device by associating the device position data to the tool's and the reaction forces that the virtual environment exerts on the tool to the forces exerted by the device. Thus for instance a Phantom can interact with the virtual environment using one, two or n proxies.
- CHAI can be used to teach basic classes in haptics due to its intuitive and easy to learn structure. Furthermore CHAI supports a *Virtual Device* that allows to move one or more points inside a virtual scene using mouse and keyboard and graphically render reaction forces (as vectors). This has proven to be a tool of great utility when teaching large haptic classes where providing one haptic device per student would be extremely expensive. CHAI has been used with a great deal of success while teaching the CS277 "Experimental Haptics" class at Stanford University in the years 2002 and 2003.
- CHAI has been steadily growing in the last three years. Due to its modular structure additional components can be easily added to CHAI. Such has

been the case for instance with the Arachi dynamic engine [3] as well as with various deformable object simulation engines such as the ones based on the Radial Elements Method [11] and the Sphere Filling Method [12]. Any new proposed addition is reviewed by a steering committee and is eventually included in the following CHAI release.

- CHAI is free and will soon be downloadable from the web sites <http://www.chai3d.org> and <http://www.hvtk.org>.
- CHAI is being used at the following institutions: Stanford University, Universidade Católica de Brasília in Brazil, the University of Siena in Italy and EPFL Institute in Lausanne, Switzerland.

3 Different types of CHAI

Two releases of CHAI will be available for download. CHAI light is the simplest and lightest implementation of CHAI, ideal for instance for a class environment. CHAI full is the full featured version of the libraries.

References

1. SensAble Technologies, <http://www.sensable.com>
2. ForcEdimension, <http://www.forcedimension.com>
3. Arachi, <http://www.arachi.com/>
4. D. Ruspini, K. Kolarov and O. Khatib: "The Haptic Display of Complex Graphical Environments" in Computer Graphics Annual Conference Series, pages 345–352, vol. 31, 1997.
5. C. Zilles and J. Salisbury: "A ConstraintBased God-Object Method For Haptic Display" in Proc. IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots, pages 146-151, vol. 3, 1995.
6. The e-touch project, <http://www.etouch3d.org/>
7. ReachIn, <http://www.reachin.se/>
8. K. Salisbury and C. Tar: "Haptic rendering of surfaces defined by implicit functions" in Proc. of ASME Dyn. Sys. and Control Div., pages 61–67, vol. 61, 1997.
9. Servotogo, <http://www.servotogo.com/>
10. Sensoray, <http://www.sensoray.com/>
11. R. Balaniuk and K. Salisbury: "Soft-tissue simulation using the Radial Elements Method ", Accepted for presentation at the IS4TM workshop, June 2003.
12. F. Conti, O. Khatib and F. Baur: "Interactive rendering of deformable objects based on a filling sphere modeling approach", Accepted for presentation at the IEEE International Conference on Robotics and Automation, May 2003.