CybHaptics: A Class for Haptics Support in Virtual Reality Systems

Daniel F. L. Souza[1], Ronei M. Moraes[2] and Liliane S. Machado[1]

[1] Departament of Informatics and [2] Departament of Statistics
Universidade Federal da Paraíba - Cidade Universitária s/n – 58051-900 – João Pessoa/PB – Brazil

danieltidus@gmail.com, ronei@de.ufpb.br, liliane@di.ufpb.br

ABSTRACT

Haptics have been largely used in VR environments, especially in training systems in which user dexterity is a mandatory requirement to determine performance. Besides that, through tactile object exploration, applications can offer a high level of interaction to users. This work presents CybHaptics, a set of classes for the CyberMed, a system for fast development of medical simulators. The goal of CybHaptics classes is to allow the use of haptics facilities in VR applications developed with the CyberMed system. These classes provide interfaces, support for several haptic devices and run in a synchronized way with other tasks, as visualization, deformation and assessment of training, in the final application.

Keywords

Haptics, VR Systems, Medical Training.

1. INTRODUCTION

VR applications have incorporated haptics in order to provide touch sensation for users [4]. These applications present specific requirements, as the simulation of procedures which require hand dexterity. With this feature is possible to provide more immersion to users and touch can be associated to visualization [2]. In medicine, several applications have been developed for training, assistance and educational purposes. These applications make computational creations (copies) of real situations in which the user can explore the environment as he/she would do in the real world.

The CyberMed system has been developed to facilitate the process of development of VR simulators for medical purposes [4]. It is composed by a set of classes which provides several facilities and support for several devices. This feature gives to the programmer a high level system which does not demand specific knowledge about devices, their programming API and specific algorithms related to VR development.

The goal of this paper is to present CybHaptics, the haptic subsystem of the CyberMed. Thus, its conception and development will be presented as well as the results obtained after its incorporation to the CyberMed.

2. HAPTIC SYSTEMS

When associated to visual displays, haptic systems greatly improve the realism of VR based simulations [1]. In this context, touch is used to explore objects features, as shape, roughness and texture [3]. The difference between haptic interaction and other interaction devices used in VR systems is the bi-directional communication feature. It gives to a single interface (device) the capability of sending and receiving information [2]. The haptic interaction happens through haptic systems, composed by the pair: device and haptic rendering routines. Most haptic devices available are sold with a control software or API. The goal of them is to provide haptic rendering techniques through a high-level programming tool for the device.

Haptic rendering can be defined as the process where haptic control routines calculate the haptic modification of a haptic scene, update this scene and send these modifications to the user in real-time [3]. Most haptic software implement a single interaction point to represent the haptic device and the contact is verified only for this point. This technique will also be used in this work for the implementation of CybHaptics.

There are several material properties which can be simulated by haptic devices. They can be divided in two groups: surface properties, as static and dynamic friction, stiffness and textures; and environment properties, as viscosity, constant forces and gravity.

3. THE CYBHAPTICS CLASSES

The CybHaptics is a set of classes dedicated to deal with haptics in the CyberMed system. Its goal is to provide an intuitive and abstract access for the integration of haptics in applications developed with CyberMed by the creation, support and management of haptic rendering tasks. In this case, the CybHaptics can be used in an independent way to create haptic scenes or can be used with other CyberMed classes to compose a VR system for medical simulation. This feature garantees the performance of the haptic rendering and its complete synchronization with other tasks (view, assess, etc).

Scene and objects topologies used by CybHaptics are acquired from CybParameters, which contains a copy of all data stored in the data structure OF. This copy is important because the CyberMed allows visual deformation and the topology of an object cannot change for the haptic rendering routines. This will happen only when an object was permanently deformed. Then, the CybParameters will update the topology from OF and the CybHaptics will start to deal with this data into the haptic rendering. Figure 1 presents the CybHaptics and its relationship with other CyberMed components.

To rendering virtual objects, the CybHaptics provide several methods to individual creation, enabling and rendering of them. Then, the user of the classes can manipulate individually an object and enable or disable it in the haptic rendering.

The calibration of haptic devices is possible through several methods and engines which monitor their activities. In this case, the user of a final application can be informed about the necessity of calibration of the device. On the other hand, the update of haptic scenes is performed through update routines, responsible by the synchronization between visual and haptic scenes. This feature offers high-level dynamic control in the construction of applications with direct manipulation of objects.

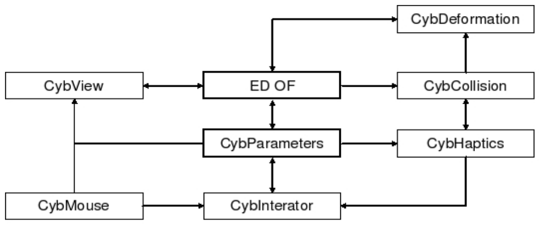

Figure 1. CybHaptics and its integration to the CyberMed system.

The attribution of material properties to objects and to the environment is also supported by CybHaptics. This is one of its main features, mostly because the development of training applications uses several and different properties to haptically describe objects. Examples of these kinds of application are simulations which allow body organs palpation or the navigation inside body structures. The process of attribution of a material property to an object is similar to the creation of a haptic object: it is necessary to create the property, enable it and after apply it in the object. This allows the individual attribution of properties for each object of the haptic scene. However, the environment or parts of it can demand haptic properties. This happen when is necessary to simulate material properties of a region inside some object. For this case, CybHaptics also offers methods to help the definition of environment properties.

4. DEVELOPMENT

The CybHaptics was developed to provide a high-level access to haptic routines, as device maintenance and haptic rendering. However, its development is flexible enough to allow low-level access for expert programmers: CybHaptics has an abstract implementation which presents features common to most of the haptic devices. To support a specific device, a programmer can inherit these features and implement them as he/she wants (Figure 2). In fact, CybHaptics hides the implementation of several classes used to support haptic devices and their functions.

In a depth view, new devices can be added to the CybHaptics by its inheritance and the implementation of the functions needed to a specific device. This implementation will not only allow the integration of the device into CybHaptics, but also to the CyberMed and all its functionalities. Figure 3 shows how this integration can be done for several haptic devices: a class HapticDeviceInterface must inherit the abstract class CybHaptics and provide and interface between several implementations and the CyberMed. Some functionalities defined by CybHaptics are: start the haptic rendering, get device position, get device rotation, get force applied, update haptic scene, get transformation matrix and idle. The last one is used to verify haptic rendering errors and/or calibration status.

The integration described below was used to provide support for the haptic devices of the Phantom family. Then, the classes used inherited CybHaptics features and implemented methods necessary to support these devices. The Figure 3 shows the diagram of classes used to integrate Phantom devices to the CyberMed.

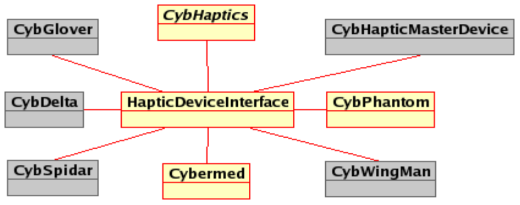

Figure 2. Integration of several devices and their relationship in CybHaptics.

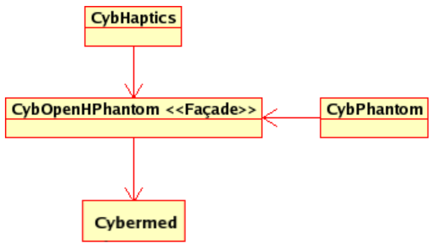

Figure 3. Classes to support Phantom devices in CybHaptics.

The CybPhatom class implements low-level access to the haptic device and uses the API provided by the device manufacturer. The haptic rendering routines and their management system are in this class. All information necessary for these routines is in the CybParameters class and will feed the haptic scene. The same CybParameters feed the visual scene.

A class CybOpenHPhantom inherits the characteristics of the abstract class CybHaptics and implements them to integrate the haptic device to the system. The CybOpenHPhantom provides a high-level control of the device routines and allows programmers an easy use of the device functions. This fact facilitates the use and programming of the haptic device and does not require previous experience or depth knowledge of the device API. Basically, the CybOpenHPhantom provides a façade between the CybPhantom and the complete system, at the same time it implements the CybPhantom features necessary to integrate the device to the system. Then, expert programmers which want to have a direct manipulation of the haptic device can override the CybOpenHPhantom and implement the functions how they desire.

The use of project pattern singleton is other interesting feature of the CybOpenHPhantom. It allow to the programmer the access to all functionalities through a single instance of the class. It avoids the creation of multiple objects to deal with the device functionalities, which could generate errors in the final application. Both CybPhantom and CybOpenHPhantom can be used by programmers who want to use Phantom haptic devices in their applications. The only experience necessary to use the high-level functions of CybHaptics is the knowledge about the creation of haptic scenes, what is similar to the process of creation of graphic scenes.

The programming language used to develop CybHaptics classes was C++. To integrate Phantom haptic devices to these classes was used the OpenHaptics API, provided by Phantom manufacturers. The developing platform was the Linux, Fedora distribution. The classes can be used with any Linux distribution and is independent of window system.

5. RESULTS

Nowadays, CybHaptics offers all the features described in section 3. It is completely integrated to the CyberMed system and is synchronized with visualization and assessment classes. This way, an application developed with the CyberMed can use the only CybHaptics in a haptic display application or can use it with other CyberMed classes to design more complex applications. In the last case, the application can integrate visualization, collision detection, deformation and assessment tasks, all synchronized and running in real time. The utilization of all potentialities of CyberMed will depends on the hardware used to run the final application.

The code below presents an example of application, which integrates several layers for a visual and haptic scene in medical training simulation, developed with the CyberMed. The goal of the application is to provide training of bone marrow harvest and includes three stages: visualization, palpation of the body and collect of material inside the body with a needle. This application provides visualization and touch.

```
#include "Cybermed/cybOpenHPhantom.h"
#include "Cybermed/cyb.h"

int main(int argc, char** argv)
{
   int numLayer = 5;
   char *arqname[30] = {"1.wrl","2.wrl","3.wrl","4.wrl", "5.wrl"};
   ofMesh<cybTraits> malha[numLayer];
   CybDataObtainer<cybTraits> data(numLayer);
   ofWrlReader<cybTraits> entrada;
   CybParameters cybCore;
   CybApplication view;

   // set haptic material properties for the skin
   interator.hapticDevice.createHapticLayers(0,true);
   interator.hapticDevice.createMaterialPropertyContext(numLayer, true);
   interator.hapticDevice.setMaterialPropertyValue(0, POPTHROUGH, 1.0f);
   interator.hapticDevice.setMaterialFace(0, POPTHROUGH, FRONT);
   interator.hapticDevice.setMaterialPropertyValue(0, DYNAMIC_FRICTION, 0.6f);
   interator.hapticDevice.setMaterialFace(0, DYNAMIC_FRICTION, FRONT);

   // set haptic material properties for the bone
   interator.hapticDevice.createHapticLayers(1,true);
  ...
   interator.hapticDevice.enableHapticMaterialProperty();

   for(int i=0; i<numLayer; i++)   {
                entrada.read(&malha[i], arqName[i]);
                data.readColor(nomeArquivo[i], i);
   }
   data.startParameters(numLayer, malha, &cybCore);
   view.setWindowName(nameWindow);
   // set object as interator
   interator.setObjectType(0, -0.020913, 0.078050, -10.034132);
   …
   view.init();
}
```

Several tests were performed with different models. The goal was to observe the refresh rate of the haptic rendering tasks. To provide a good simualtion, the refresh rate should be around 1000 Hz. This rate offers a complete absence of delays in the haptic display. Table I shows the refresh rate of the haptic scene observed during the interation with different models. The values were collected in an application with visual and haptic display. The tests were performed in a Athlon XP 2600 processor, in a PC with 1Gb RAM, 256Mb graphic card, in a Linux Fedora Core 4 32 bits operational system.

Table I. Refresh rate of the haptic scene whit different models.

Model	Number of Vertices	Number of Triangles	Refresh rate of haptic scene (in Hertz)
Sphere	993	1983	~2333
Skin	3367	5534	~2000
Bone Marrow	8036	16072	~1550
Illiac Bone	12070	24164	~1330

6. CONCLUSIONS

This paper presented CybHaptics, a set of classes developed for the CyberMed system. CybHaptics provides interfaces, support for several haptic devices and is completely synchronized with other tasks in a VR system, as visualization, deformation and assessment of training.

The design of the classes allows a high-level use of haptic functions for programmers and leaves the details of haptic rendering implementations internal to the system. However, low-level access to haptic functions are also available and can be used by expert programmers. This feature of the CybHaptics gives flexibility to the development of applications which integrates haptic functionalities.

At this moment CybHaptics is completely integrated to the CyberMed system with support to the haptic devices of the Phantom family. Due to the design of the classes, the support of new devices can be easily incorporated. New releases of the classes intend to include the support of new haptic systems and the simultaneous use of several haptic devices in an application.

7. ACKNOWLEDGMENTS

This work is supported by Brazilian Council for Scientific and Technological Development, (CNPq CT-INFO 506480/2004-6) and Brazilian Research and Projects Financing, (FINEP 01-04-1054-000).

8. REFERENCES

[1] Cagatay, B., Haptic Rendering, Tutorial online: http://network.ku.edu.tr/~cbasdogan/tutorials/haptic_tutorial.html. Access in October, 2005.

[2] Salisbury, K., Conti, F., Barbagli, F. Haptic Rendering: Introductory Concepts. *IEEE CG&A*, April, 2004, 24-32.

[3] Salisbury, K. et al., Haptic Rendering: Programming Touch Interaction with Virtual Objects. *Proc. ACM 1995 Symposium on Interactive 3D Graphics*, Monterey CA, April, 1995, 146-151.

[4] Sherman, W., R. Understanding Virtual Reality: Interface, Application, and Design, Morgan Kaufmann Pub, CA, 2003.