Function-based haptic interaction in cyberworlds

Lei Wei, Alexei Sourin, Olga Sourina
Nanyang Technological University, Singapore
weil0004 | assourin | eosourina @ntu.edu.sg

Abstract

We seek to further expand the shared collaborative potential of cyberworlds by using haptic force-feedback in shared virtual scenes. We propose how to define density of the objects, together with their geometry and appearance, by using mathematical functions. We illustrate this concept by developing software which allows us to touch and feel surfaces of VRML and X3D objects, convert them to solid objects as well as create any other solid objects using the function-based extension of VRML and X3D. We define geometry, appearance and density of the solid objects by implicit, explicit and parametric functions straight in the VRML/X3D code or in dynamic-link libraries. Since the function-based models are small in size, it is possible to perform their collaborative interactive modifications with concurrent synchronous visualization at each client computer with any required level of detail. We illustrate the proposed with several application examples.

1. Introduction

We seek to further expand the shared collaborative potential of cyberworlds by using haptic force-feedback in shared virtual scenes. Until now, haptic devices were rather exotic and usually expensive gadgets. However with the arrival on the consumer market of affordable interactive 3D touch devices like Novint Falcon [http://home.novint.com], we anticipate that haptic communication will become as common as interaction with mice and joysticks. This assumes that there must be developed software tools which can be used for haptic interaction with surfaces and interiors of the virtual objects in commonly available data formats, as well as generic and intuitive ways of creating such solid objects.

In this paper we propose to define density of the objects, together with their geometry and appearance, by using mathematical functions. We illustrate this concept by developing software which allows us to touch and feel surfaces of VRML and X3D objects [http://www.web3d.org], convert them to solid objects as well as create any other solid objects using the function-based extension of VRML and X3D. We also show how to perform web-based haptic collaboration with geometric objects defined using standard VRML and X3D as well as their extension FVRML/FX3D. We found a way how to use a commonly available communication platform for doing it.

In the rest of this Section we provide a brief survey of networked haptic interaction and an introduction of FVRML/FX3D. In Section 2 we discuss the design and implementation of the haptic plug-in for VRML and X3D and the ways of defining collision detection in the scenes. Finally in Section 3 we give examples of using the developed software and summarize the work in the Conclusion.

1.1 Networked haptic interaction

Haptic technology provides an interface to the user via the sense of touch by applying forces, vibrations and/or motions to the user. There are many varieties of such devices. In this paper we only consider those which are implemented as a haptic stylus on a robotic arm allowing for navigating a virtual instrument in 3D space with 6 degrees of freedom and variable force-feedback. The examples of such devices are those produced by Sensable Technology, Immersion, Force Dimension, Quanser, MPB Technologies, ERGOS Technologies, and Novint.

We identify three approaches to what could be called networked haptic rendering.

The first approach assumes making stand-alone haptic applications that are capable of working with 3D web data. Thus H3D API [http://www.h3d.org] can directly load and parse X3D files, and then render the scene both graphically and haptically. It utilizes OpenGL for graphical rendering and Sensable OpenHaptics Toolkit for force rendering. In other work [1], Sensable GHOST SDK is used to implement a web

based plug-in for haptic device. The plug-in makes use of the VRML file reader included in the GHOST SDK. Distributed haptic museums, discussed in [2] and [3], allow the visitors to explore 3D items with haptic devices.

Making plug-ins to web browsers is another way to implement web-based haptic applications. Compared with the first approach, these applications do not have an executable program, but rather an ActiveX control or a dynamic linking library file called by the web browser. Thus, the open source CHAI3D [http://www.chai3d.org] is a set of C++ libraries for haptic interaction and visualization. It provides integration with ActiveX for web-embedded haptic applications. In [4], a haptic tele-rehabilitation system is described, in which force feedback is incorporated into a web page using a Java Applet. Immersion Corporation's FEELtheWEB ActiveX browser control is used in this project to manage the force-feedback joystick and receive function calls through HTML. In [5] haptic force rendering on the web, based on using CHAI3D and XVR 3D web application platform, is implemented by adopting *.aam* file format as the model format and *.s3d* as the scene format, which are compiled and deployed as an output binary file to an HTML web page.

In the third approach, haptic collaborative applications are built so that the connection between the collaborating parties is established directly by TCP/UDP protocol. There are quite a few works in this area. The architecture of such applications is based on server-client and peer-to-peer configurations as well as on their mixtures. Some applications use common data formats like VRML, X3D, and CAD (e.g., [6, 7, 8, 9]), while other authors introduce their own formats for defining 3D objects and scenes (e.g., [10, 11, 12, 13, 14, 15, 16]).

1.2 Function-based web visualisation: FVRML/FX3D

Hybrid function-based shape modeling with application to VRML and X3D was introduced in [17] and further reported in [18, 19, 20, 21]. It allows for defining time-dependent geometric objects, their appearances and transformations by parametric, implicit and explicit functions which can be used concurrently.

Implicit and explicit functions used in the extension are functions of 3D coordinates and the time. Implicit functions define surfaces. Explicit function values can be used either as arguments of other functions or as indicators of the sampled point locations for defining bounded solid objects in FRep [22] sense. In this case, the function equals to zero for the points located on the surface of an object. Positive values indicate points inside the object and negative values are for the points outside the object. Explicit functions can also define transparency, shininess, ambient intensity and color of objects as well as displacement values of the 3D textures. Parametric functions, as used in the extension, define curves, surfaces and solid objects. Parametric functions can also define components of diffuse, specular and emissive colors.

The syntax of the FVRML/FX3D extension [http://www.ntu.edu.sg/home/assourin/fvrml.htm] is similar to that of VRML and X3D. The functions are either typed as individual formulas and java-style function scripts or defined as external library functions (e.g., 3D reconstruction functions). FVRML/FX3D opens VRML and X3D to practically any type of geometry, 3D colors and geometric textures. It also introduces to VRML and X3D set-theoretic operations (union, intersection, difference), as well as allows for defining any other sophisticated operations (e.g., geometric and color metamorphoses). The function-defined objects can be used together with the common VRML and X3D objects and appearances, as well as allow for using the common object and appearance fields as their parts.

2. Haptic interaction in VRML and X3D scenes

In many practical applications there is a need in manipulating objects in 3D scenes defined by VRML and X3D. Desktop haptic devices naturally make these manipulations easier for the users since 3D rotations and translations can be conveniently defined by motion of the device stylus. Besides this, there can be a need to feel the surface of the objects which are in fact defined by polygons. However, to the best of our knowledge, there is no generic haptic extension of VRML and X3D available.

We considered as a first application task the development of such haptic plug-in that will work with VRML/X3D browsers and will be transparent to the applications which do not require haptic devices.

As a second task, there can be a need to convert standard VRML/X3D objects, confined by closed polygonal surfaces, into solid objects with variable density to be able to examine them with haptic devices. This requires us to introduce a new functionality to VRML and X3D to define the density for converting bounded objects into solids for haptic rendering, while the graphics rendering still remains polygon-based.

Finally, one should be able to define solid objects, which can not be defined by standard functions of

VRML and X3D, while still staying within the visualization pipeline of VRML/X3D. For example, there can be medical applications requiring reconstruction of 3D solid objects from MRI data followed by their haptic examination.

The second and third tasks can be efficiently implemented by introducing a function-based density node to FVRML/FX3D mentioned in 1.2. The VRML node diagram for the modified function-defined shape node *FShape* will then look as it is shown in Figure 1.

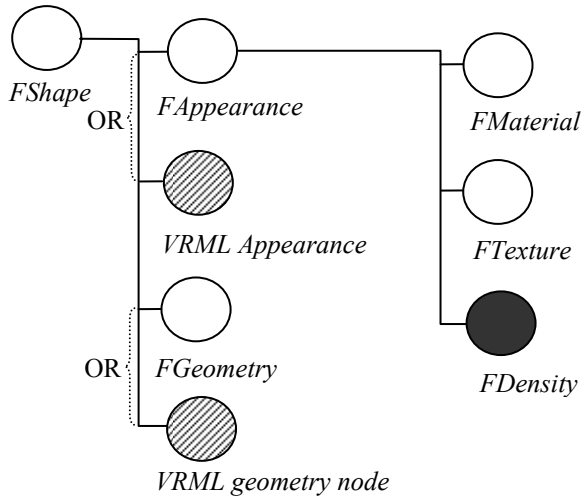


Figure 1. FVRML/FX3D node diagram including standard VRML/X3D nodes (hatched) and the introduced density node (shaded)

FShape node, like *Shape* node of VRML and X3D, is a container for either regular VRML/X3D or function-defined geometry and appearance nodes. We just add to *FAppearance* node a new function-defined density node *FDensity*. Since the nodes of FVRML/FX3D can be used together with the standard VRML and X3D nodes, we can then use *FDensity* with common bounded VRML/X3D geometric objects, which thus will be converted to solid objects, as well as with the geometric solid objects defined by functions in *FGeometry* node. The density can be defined in a way similar to how 3D colors can be defined functionally—by individual explicit functions, by function scripts, or by using dynamic-link library functions (e.g., trilinear interpolation of densities defined by voxels in MRI data file) that can be called by the VRML/X3D browser from the VRML/X3D source code.

Hence, with reference to VRML, to convert a common bounded VRML object defined by polygons into a solid object with a uniform density, the following code has to be written:

```
shape FShape{
    appearance FAppearance{
        material FMaterial{ }
        density FDensity {definition "1"}}
        geometry IndexedFaceSet {
            VRML object defined by polygons ...}  }
```

To replace a standard VRML node with an FVRML solid object, we have to replace the polygon-based geometry node *IndexedFaceSet* with the function-defined geometry node *FGeometry*. In the following code, it is a sphere distorted by a solid noise which is defined by an explicit function. The solid has a function-defined density with a harder core and softer peripheral areas:

```
shape FShape{                                  FVRML density node
  appearance FAppearance{
   material FMaterial{ }
     density FDensity {definition "1-x^2-y^2-z^2"}}
                                               FVRML solid geometry
  geometry FGeometry {
     definition "0.25 – x^2 – y^2 – z^2
    +0.03*(sin(12*atan2(x,z+0.04*sin(y*25)))-0.7)"}
  }
```

Finally, to reconstruct a 3D solid with 3D color and density from some MRI data, the following FVRML code will be used:

```
shape FShape{
 appearance FAppearance{
    material FMaterial{                        3D color interpolated
    diffuseColor "                             from MRI data
function frep(x,y,z){
fun=trilinear(x,y,z/1.1,240,256,160,'MRI.dat');
patternValue(fun);}"
patternKey    [0  16 18 55 255 ]
patternColor [1 1 1  1 0.9 0.8  1 0.9 0.8  1 1 1  1 1 1]
    }
    density FDensity {                         Density
definition "function frep(x,y,z){
fun=(trilinear(x,y,z,240,256,160,'MRI.dat') - 9)/255;
return fun;}"}
 }
    geometry FGeometry {                       Geometry
definition "function frep(x,y,z){
fun=(trilinear(x,y,z,240,256,160,'MRI.dat') - 9)/255;
return fun;}" }    }
```

Here, the geometry, color and density are defined independently to finally merge into one solid object. The object definition uses external function *trilinear*, which is written by the user and put into the dedicated

folder as a *.dll* file. It is a function of the Cartesian coordinates *x,y,z* and the MRI data as a parameter. The MRI data file can be either stored on the hard drive or on a web server.

FX3D definitions can be done in a similar way. We give an example of FX3D code in Section 3.

Since, compared to the polygon-based VRML/X3D models, the function-based models are small in size, it is possible to perform their rapid exchange across the Internet for making collaborative interactive modifications with concurrent synchronous visualization at each client computer with any required level of detail. It is also possible to use the function-based models together with large data sets coming from CT and MRI scanners. In that case only the modifications to the models can be exchanged while the original data are kept on the client computers or shared on a web server.

To solve the first task—allowing for haptic rendering of regular VRML/X3D scenes—we designed and implemented a plug-in which works with blaxxun Contact and Bitmanagement BS Contact VRML/X3D browsers. The architecture of the plug-in is shown in Fig 2.

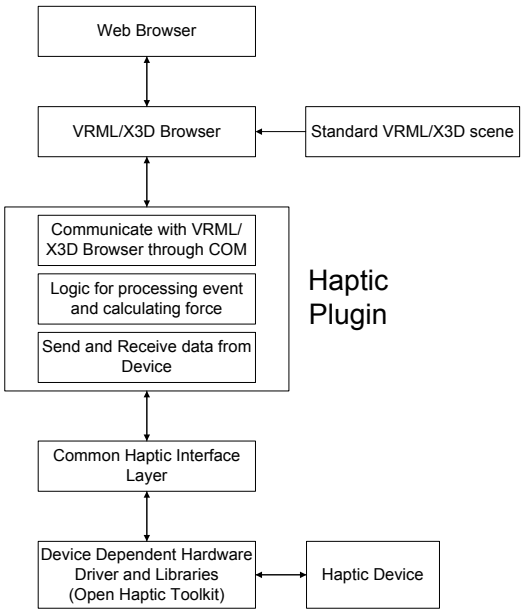


Figure 2. Architecture of the haptic plug-in

At the low level we placed the Device Dependent layer consisting of the hardware drivers and device libraries, which build communications between the computer and the device. We used Sensable OpenHaptic Toolkit with several Sensable Phantom haptic drivers installed (Phantom Omni, Desktop, and Premium 6DOF). The foundation layer of the OpenHaptic Toolkit is the Haptic Device API (HDAPI). It enables users to render forces directly and to control low-level runtime details through the driver. Haptic Library API (HLAPI) is a higher and simpler layer that builds on HDAPI and resorts to OpenGL for graphical rendering. We use HDAPI for more detailed and low-level control of the device behavior. Above the Device Dependent layer there is the Common Haptic Interface Layer, which is a wrapper layer to handle different drivers and libraries from different vendors, and to expose a group of commonly used interfaces to the developer. By introducing this layer, we make the rest of the plug-in device independent.

The core part of the haptic plug-in first reads the actual position of the device and maps it into the virtual environment as haptic interface point (HIP) position. Then it will monitor the virtual position of the haptic device stylus and continuously check whether it collides with objects in the scene at the haptic frequency 1000HZ. Since this haptic loop is much faster than the visualization loop, we use asynchronous scheduler between these two threads. When a collision occurs between the virtual stylus and the objects in the scene, the VRML/X3D browser will send the corresponding data, such as *Hit Point* and *Hit Normal*, back to the plug-in through the COM interface provided by BS Contact SDK. The plug-in will calculate the corresponding force vector according to our collision detection algorithm and will make the necessary coordinate transformations. After that, the plug-in sends the force vector to the haptic device and applies force rendering. This process continues at a high refresh rate to ensure the output force is smooth and stable. No changes to the VRML/X3D scene are required except a link (inline call) to a small VRML/X3D file which performs interfacing with the core function of the plug-in.

The collision algorithm of the core part consists of two parts: collision with the surface and motion inside solid objects.

To compute collision detection with surfaces of the objects we use *computeRayHit* method provided by Bitmanagement SDK. However the direct application of this method in the scenes with large number of polygons is not feasible since the update rate of the haptic device is much higher than that of the visualization loop which controls the *computeRayHit* method execution. Therefore in our algorithm, with the visualization frequency we use *computeRayHit* to detect coordinates (x_0, y_0, z_0) of the intersection surface contact point (SCP) on the object surface with a ray cast from the HIP towards the scene in the direction of the haptic stylus. After that, we use an equation of the plane which is close to the actual surface:

$$g = a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$$

where (a, b, c) are coordinates of the normal **N** built at the SCP. If $g>0$, the HIP is still above the plane and there is no collision with the object. If $g\leq 0$, the HIP is below the plane and inside the object, and we calculate the distance between the HIP and the plane

$$D = |ax + by + cz - (ax_0 + by_0 + cz_0)| / \sqrt{a^2 + b^2 + c^2}$$

and the feedback force according to the Hook's law

$$\mathbf{F} = D\mathbf{N}.$$

These calculations are performed with the high haptic rendering frequency which ensures smooth interpolation of the force, however we improved it even further by doing multiple predictions of HIP motion in 26 different directions, out of which we select the direction with the minimum distance between the SCP and HIP.

Other principles are used when the HIP is moving inside the object. We use the concept of velocity for defining the force direction. The force will always generate in the direction opposite to the HIP movement and proportional to the density. We obtain the velocity of HIP from the OpenHaptic Toolkit at the haptic frequency and interpolate it as follows:

$$V_{current\ interpolated} = C_1 V_{previous} + C_2 V_{current}$$

where $V_{previous}$ is the velocity at 1/1000 sec before the current time and C_1 and C_2 are some interpolation coefficients.

The feedback force is then calculated as follows:

$$F = V_{current\ interpolated} L$$

where L is the function-defined density evaluated at HIP.

The introduction of the density field required both changes to the FVRML/FX3D plug-in and the haptic plug-in discussed above. For the haptic plug-in architecture we just added a new node which needs to be interpreted by the function parsers that are integrated into the plug-in (Fig. 3).

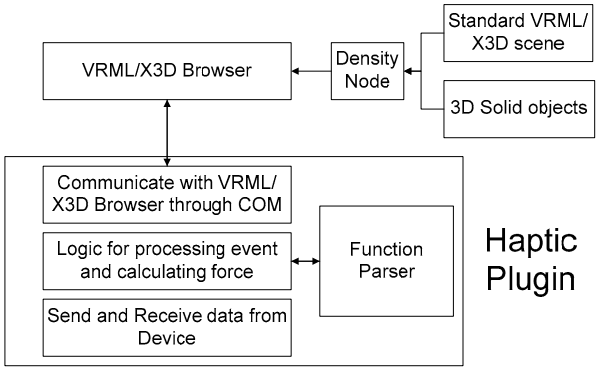


Figure 3. Changes to the architecture of the haptic plug-in by including FVRML/FX3D

When the plug-in runs, the VRML/X3D browser will also send the function expression to the plug-in through the COM interface. The function is evaluated in the plug-in and the result is applied to express the interior density of the point which the virtual stylus is touching. A force vector will then be evaluated and sent out to the haptic device for rendering.

3. Application examples

In Fig. 4, a snapshot of a VRML scene explored with a haptic device is shown. A 3D area which can be haptically rendered is indicated with a semi-transparent box that is displayed in front of the observer. It becomes visible when the haptic rendering is activated by the user. The surfaces of the object can be touched with a 3D crosshair cursor displayed in 3D scene.

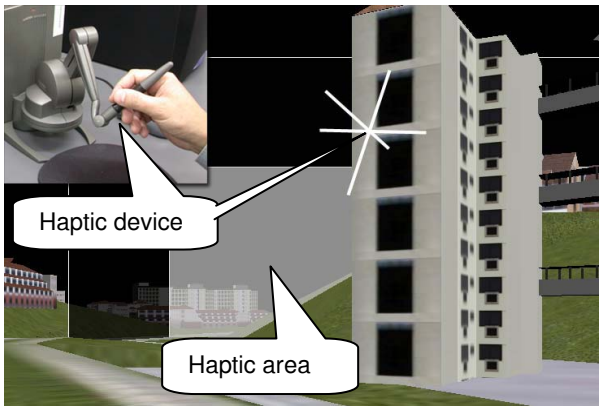


Figure 4. Exploring a standard VRML scene with a desktop haptic device

In Fig. 5, a standard X3D Indexed Face object is converted into a solid object and explored with a haptic device. The area available for haptic rendering is reduced by the user to increase the precision of the force-feedback

In Fig. 6, an example of a collaborative haptic shape modeling session is shown. Since there is no native support for collaboration in VRML and X3D, a third party communication platform must be obtained to develop such modeling tools. We did not have an aim to develop our own communication software but rather looked for an available communication platform which could be used for our purposes. After analyzing different options, we selected *blaxxun Communication Server*. It is also possible to make it work together with *Bitmanagement BS Contact VRML/X3D* browser.

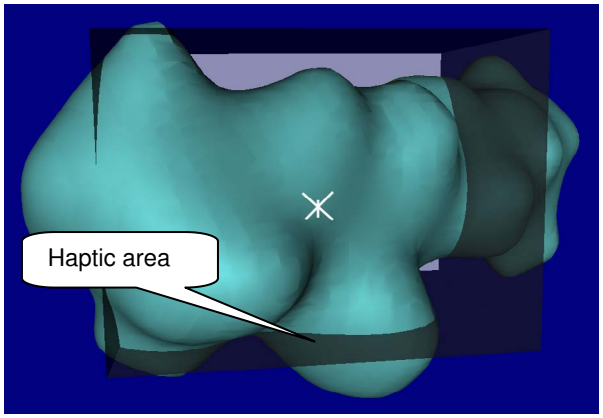
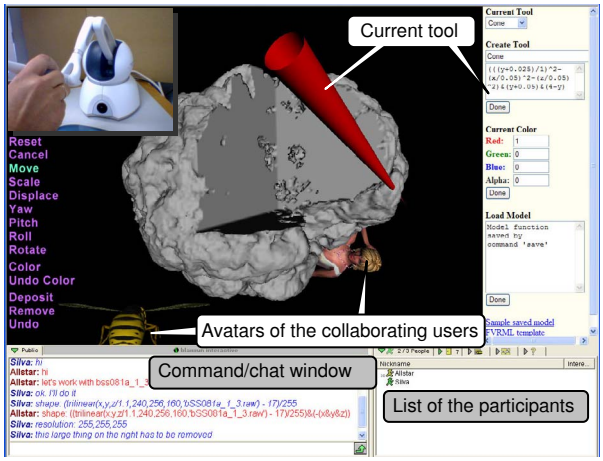
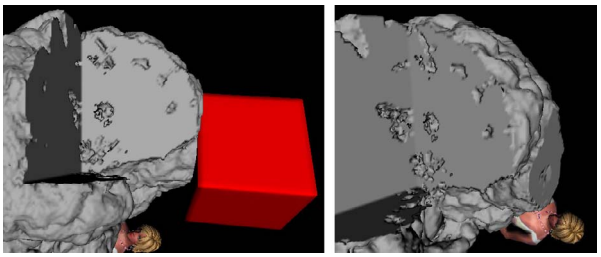


Figure 5. An X3D object is converted to a solid with a uniform density



(a)



(b) (c)

Figure 6. Collaborative haptic examination and refining of the human brain model reconstructed from MRI data

In the user interface of this application there are three parts which are the shared 3D collaboration scene where haptic rendering is allowed (top-left), the control panel (top-right) and the command/chat windows (bottom).

The 3D haptic modeling scene and the chat area are shared among the users in the same session, while the control panel is user-specific and not shared. The users can type commands as well as chat in the command/chat pane concurrently. To change the current tool, the pre-defined tools and existing customized tools can be chosen from the drop-down menu in the control panel.

A 3D human brain model in Fig. 6a is reconstructed from MRI data. This is done by the user '*Silva*' who enters a command requesting a trilinear interpolation of the shape, after that, another user '*Allstar*' issues a new command where a part of the interpolated shape is cut away with set-theoretic subtraction operations. Next, the first user increases the rendering resolution of the shape. Interactive tools for haptic rendering of the shape and its modifications can be either selected from the given set or defined on the spot analytically. In Fig. 6a, this is a red cone that is used for haptic probing of the internal structure of the brain model. Its geometry is defined by an implicit function in the '*Create Tool*' interactive box.

The tools can be also used for changing the shape's geometry and appearance. To apply an interactive tool, the user clicks at any point on the object being modeled. The 3D object representing the current tool will be placed to the scene at the selected point. The size, position and orientation of the tool can be changed interactively either with the haptic device or with a common mouse. After placing the tool, the user selects one of the pre-defined operations to modify the geometry or the color of the object. In the application which is shown in Figure 6, three such operations are defined: removing of material, depositing of material and applying color. Removing material subtracts the object representing the tool from the current object. Depositing material unifies the tool with the current object. Coloring operation blends the color assigned to the tool with the current 3D color of the object according to the transparency assigned to the tool. The color blending is performed only at the part of the current object that intersects with the tool. When the tool is not transparent, the color of the tool overrides the color of the current object at the application area. In Fig 6 b,c the artifact on the brain model is removed by cutting it away with a box tool.

In this collaborative session, only function-defined modifications to the original MRI model are exchanged over the Internet. The size of such models does not exceed a few kilobytes, while the original MRI data file with a size of 10 MByte is shared on a web server. The object being modeled can be eventually saved in VRML or X3D code. The example of an exported X3D code of this scene is given in Fig. 7.

```
<ProtoInstance name="FShape">
<fieldValue name="bboxSize" value="5 5 5" />
<fieldValue name="bboxCenter" value="0 0 0" />
<fieldValue name="appearance">
<ProtoInstance name="FAppearance">

<fieldValue name="material">
<ProtoInstance name="FMaterial">
      <fieldValue name="diffuseColor"
      value="function frep(x1,y1,z1,t){
      r=0.7;g=0.7;b=0.7;
      return 0;}" />
</ProtoInstance>                    Color definition
</fieldValue>

<fieldValue name="density">
<ProtoInstance name="FDensity">
<fieldValue name="definition"
      value="function frep(x1,y1,z1,t){
      x=x1; y=y1; z=z1;
      fun=((trilinear(x,y,z/1.1,240,256,160,
      'http://www…/brain.dat')-9)/255));
      return fun;}" />
</ProtoInstance>                    Density definition
</fieldValue>

</ProtoInstance>
</fieldValue>

<fieldValue name="geometry">
<ProtoInstance name="FGeometry">
 <fieldValue name="resolution"
            value="100 100 100" />
 <fieldValue name="definition"
      value="function frep(x1,y1,z1,t){
      x=x1;y=y1;z=z1;
      fun=((trilinear(x,y,z/1.1,240,256,160,
      'http://www…/brain.dat')-9)/255)&amp;amp;
      (-(x&amp;amp;y&amp;amp;z));
  x=x1*(-0.1)+y1*(-0.2)+z1*(-0.3)+(-0.1);
  y=x1*(0.3)+y1*(-0.1)+z1*(-0.1)+(-0.3);
  z=x1*(0.01)+y1*(-0.3)+z1*(0.18)+(0.1);
  fun=fun&amp;amp;(-((x+0.1)&amp;amp;
   (0.1-x)&amp;amp;(y+0.1)&amp;amp;
   (0.1-y)&amp;amp;(z+0.1)&amp;amp;(0.1-z)));
      return fun;}" />
 </ProtoInstance>                   Geometry definition
</fieldValue>

</ProtoInstance>
```

Figure 7. Exported FX3D code

In the geometry definition part of the code, the function script contains a link to the web-located MRI data file which is used as a data for the trilinear interpolation function. The script also defines set-theoretic operations (subtractions) which were done in the command mode (removing a part of the shape) and performed interactively by one of the participants (removing an artifact by the box tool). The density of the shape is also defined by applying the trilinear interpolation function which uses as data the original web-located MRI file.

4. Conclusion

In this paper we report on our approach to haptic collaborative modeling in shared virtual scenes. We proposed to define density of the objects, together with their geometry and appearance, by using implicit, explicit and parametric functions. We illustrated this concept by developing software which allows us to touch and feel surfaces of VRML and X3D objects, convert them to solid objects as well as create any other virtual solid objects using the function-based extension of VRML and X3D.

We implemented a haptic plug-in to two VRML and X3D browsers (blaxxun contact and Bitmanagement BS contact VRML/X3D). We described the architecture of the plug-in and the main collision detection algorithm. Still staying within the VRML/X3D rendering pipeline designed for objects built from polygons, we are able to perform haptic rendering of shared VRML and X3D scenes by touching surfaces of the objects with different desktop haptic devices. Besides this, VRML/X3D objects can be converted to solid objects by defining their inner densities with mathematical functions. This can be done by typing analytical functions straight in the VRML/X3D code or by defining them in dynamic-link libraries. Geometric solid objects can be also defined by functions. We allow for reconstructing solid objects from CT and MRI data. Since the function-defined models are small in size, we can efficiently use them in web-based collaborative projects.

5. Acknowledgements

This project is supported by SBIC Innovative Grant RP C-012/2006 “Improving Measurement Accuracy of Magnetic Resonance Brain Images to Support Change Detection in Large Cohort Studies” and MOE grant RG10/06 “Visual and Force Feedback Simulation in Nanoengineering and Application to Docking of Transmembrane a-Helices”.

5. References

[1] M. O'Malley, S. Hughes, "Simplified Authoring of 3D Haptic Content for the World Wide Web", Proc. *11th Symp on Haptic Interfaces for Virt Env and Teleoperator Systems,* 2003, pp. 428-429.

[2] T. Asano, Y. Ishibashi, S. Minezawa, M. Fujimoto, "Surveys of Exhibition Planners and Visitors about a Distributed Haptic Museum", Proc. *2005 ACM SIGCHI Int Conf on Advances in Computer Entertainment Technology ACE '05*, ACM, 2005, pp. 246-249.

[3] M.L. McLaughlin, G. Sukhatme, C. Shahabi, "The Haptic Museum", Proc. *EVA 2000 Conf on Electronic Imaging and the Visual Arts*, 2000.

[4] D.J. Reinkensmeyer, C.T. Pang, J.A. Nessler, "Painter CC. Java Therapy: Web-Based Robotic Rehabilitation", Proc. *7th. Int Conf on Rehabilitation Robotics*, 2001, pp. 66-71.

[5] E. Ruffaldi, A. Frisoli, M. Bergamasco, C. Gottlieb, F. Tecchia, "A Haptic Toolkit for the Development of Immersive and Web-Enabled Games", Proc. *ACM Symp on Virtual Reality Software and Technology*, ACM, 2006. pp. 320-323.

[6] A. Hamam, S. Nourian, R. Naim El-Far, F. Malric, X. Shen, N.D. Georganas, "A Distributed, Collaborative and Haptic-Enabled Eye Cataract Surgery Application with a User Interface on Desktop, Stereo Desktop and Immersive Displays", Proc. *IEEE Int w/s on Haptic Audio Visual Environments and their Applications*, *HAVE 2006*, IEEE, 2006, pp. 105-110.

[7] X. Shen, J. Zhou, A. Saddik, N.D. Georganas, "Architecture and Evaluation of Tele-Haptic Environments", Proc. *8th IEEE Int Symp on Distributed Simulation and Real Time Applications*, IEEE DS-RT 2004, 2004, pp.53-60.

[8] R. Iglesias, E. Prada, A. Uribe, A. Garcia-Alonso, S. Casado, T. Gutierrez, "Assembly Simulation on Collaborative Haptic Virtual Environments", Proc. *15-th Int Conf in Central Europe on Comp Graphics, Visualization and Comp Vision, WSCG'07*, 2007, pp. 241-247.

[9] C. Basdogan, C.H. Ho, M. Slater, M. Srinavasan, "An Experimental Study on the Role of Touch in Shared Virtual Environments", *ACM Transactions on Computer-Human Interaction*, 7(4), ACM, 2000, pp. 443-460.

[10] M.Y. Sung, Y. Yoo, K. Jun, N.J. Kim, J. Chae, "Experiments for a Collaborative Haptic Virtual Reality", Proc. *16th Int Conf on Artificial Reality and Telexistence, ICAT'06*, 2006, pp. 174-179.

[11] D.J. Hubbold, "Collaborative Stretcher Carrying: A Case Study", In Proc. *8th Eurographics Workshop on Virtual Environments*, Eurographics Press, 2002, pp. 7-12.

[12] M.Q. Alhalabi, S. Horiguchi, "Tele-Handshake: A Cooperative Shared Haptic Virtual Environment", Proc. *EuroHaptics 2001*, Eurographics Press, UK, 2001, pp. 60-64.

[13] K. Hikichi, I. Arimoto, H. Morino, K. Sezaki, Y. Yasuda, "Evaluation of Adaptation Control for Haptics Collaboration over the Internet", Proc *IEEE Communications Quality & Reliability Int Workshop*, IEEE, 2002. p.218-222.

[14] D. Morris, N. Joshi, K. Salisbury, "Haptic Battle Pong: High-Degree-of-Freedom Haptics in a Multiplayer Gaming Environment", Proc. *Experimental Gameplay Workshop, GDC 2004*, 2004.

[15] I. Goncharenko, M. Svinin, S. Matsumoto, Y. Masui, Y. Kanou, S. Hosoe, "Cooperative Control with Haptic Visualization in Shared Virtual Environments", Proc *8th. Int Conf on Information Visualization, IV04*, 2004, pp. 533-538

[16] I. Fukuda, S. Matsumoto, "A Robust System for Haptic Collaboration over the Network", Proc. *Touch in Virtual Environments Conference*, University of Southern California, 2002.

[17] F.M. Lai, A. Sourin, "Function-defined Shape Node for VRML", Proc. *Eurographics 2002, Short Presentations*, Eurographics Press, 2002, p. 207-215.

[18] Q. Liu, A. Sourin, "Function-based Representation of Complex Geometry and Appearance", Proc. *ACM Web3d 2005*, ACM Press, 2005, pp. 123-134.

[19] Q. Liu, A. Sourin, "Function-based Shape Modeling and Visualization in X3D", Proc. *ACM Web3D 2006*, ACM Press, 2006, pp. 131-11.

[20] Q. Liu, A. Sourin, "Function-based Shape Modelling Extension of the Virtual Reality Modelling Language", *Computers & Graphics*, Elsevier, 30(4), 2006, pp. 629-645.

[21] Q. Liu, A. Sourin, "Function-defined Shape Metamorphoses in Visual Cyberworlds", *The Visual Computer*, Springer, 22(12), 2006, pp. 977-990.

[22] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, "Function Representation in Geometric Modeling: Concepts, Implementations and Applications", *The Visual Computer*, Springer, 11(8), 1995, pp. 429-446.