FUNCTION-BASED VISUAL AND HAPTIC RENDERING,
INTERACTION AND COLLABORATION IN SHARED VIRTUAL SPACES

NANYANG TECHNOLOGICAL UNIVERSITY

FUNCTION-BASED VISUAL AND HAPTIC RENDERING, INTERACTION AND COLLABORATION IN SHARED VIRTUAL SPACES

LEI WEI

LEI WEI

SCHOOL OF COMPUTER ENGINEERING

2011

2011

FUNCTION-BASED VISUAL AND HAPTIC RENDERING, INTERACTION AND COLLABORATION IN SHARED VIRTUAL SPACES

LEI WEI

LEI WEI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2011

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by myself and has not been submitted for a higher degree to any other University or Institution.

…… ……

Date Signature

Abstract

Current collaborative shape modeling techniques could not utilize all our senses to achieve maximum immersion. Visual rendering alone is unable to convey object properties other than geometry and appearance, such as physical properties. This causes difficulties in practical modeling, and it could be solved by incorporating haptic rendering. On the other hand, traditional polygon-based modeling representation yields large files, which conflict with the limited network bandwidth in collaborative shape modeling. This deficiency could be overcome by employing memory-efficient function-based representations. A survey of the web visualization and shared virtual environments methods and tools leads to the conclusion on the active research niche for proposing new data models and frameworks suitable for visual and haptic rendering in shared virtual spaces.

A novel uniform modeling paradigm of defining virtual objects' geometry, visual appearance and tangible physical properties is therefore proposed, in which these three entities are defined in their own coordinate domains and then merged into virtual objects in the application problem coordinate domain. To provide for a faster model exchange and any precision of the representation, mathematical functions and procedures are used for defining geometry, appearance and physical properties.

An innovative visual and haptic collaborative framework for shared virtual spaces is also proposed, where the visual and haptic pipelines complement each other to provide a simple and efficient solution to problems requiring collaboration on the web. Mathematical functions and procedures are also adopted there to provide for rapid information transmission and flexible interactive modeling operations.

In order to validate the research niche for the proposed modeling paradigm and collaborative framework, the corresponding core algorithms and software tools are implemented as a function-based extension of two ISO standards -- Virtual Reality Modeling Language (VRML) and its current successor Extensible 3D (X3D). In the virtual objects of the extended VRML and X3D, geometry, appearance and physical properties (surface friction and tension, density and force fields) are defined by mathematical functions and algorithmic procedures. These additional objects can be used on their own as well as in combination with the standard object definitions of VRML and X3D. Haptic collision detection and rendering algorithms are implemented to identify the location of the haptic tool in the scenes as well as to consistently and seamlessly determine physical properties when the haptic tool moves in the scenes with objects having different sizes, locations, and mutual penetrations. Two frameworks based on the strong and thin server concepts are implemented to illustrate practical collaboration potentials. Core concepts of shared objects, virtual tools and shared events are defined. Algorithms for information transmission, synchronization, and consistency control are devised to provide reliable and fast interaction among different client computers.

Last but not least, several comprehensive examples and practical collaborative applications are presented and discussed to verify the development and practical value of the proposed ideas.

Acknowledgements

First of all, I would like to thank my supervisor Associate Professor Alexei Sourin for his invaluable instructions. I am grateful for his help throughout my graduate study in Nanyang Technological University. He helps me to understand basic ideas, and to correct theoretical findings. Without his effects, ideas and comments, the project could not reach this point.

I would also like to thank Nanyang Technological University and School of Computer Engineering for providing me the opportunity and financial support to further my study and research. Special thanks go to Centre for Advanced Media Technology for providing me the first class equipment and technical supports.

Last but not least, I would like to thank my friends and colleagues, Dr. Qi Liu and Dr. Konstantin Levinski, for their precious support. Furthermore, I wish to thank my parents, their selfless concern and encouragement are the best support during my postgraduate study.

Contents

List of Figures

Chapter 1 Introduction

In this chapter, a brief introduction to the thesis is given. In Section 1.1, the motivation of the project is presented, and the research niche is discussed. Section 1.2 defines the objectives this thesis going to achieve. Brief explanation of the approaches proposed in this thesis is given in Section 1.3. Finally in Section 1.4 the organization of this thesis is explained.

1.1 Motivation

Collaborative shape modeling has been playing a vanguard role in computer graphics for years. Due to the emergence of personal computer and Internet, the ability to use all our senses for achieving maximum immersion into shape modeling is widely studied. Researchers realize that visual rendering alone is unable to convey object properties other than geometry and appearance, such as physical properties. This deficiency causes difficulties in practical modeling and therefore interests start to emerge in introducing haptic interaction into shape modeling to better represent various object properties.

Haptic interaction is one of the fast developing research areas in Computer Graphics. While research in visual rendering is quite mature, haptic rendering is still a rather exotic research and development issue. However, due to the arrival on the market of affordable haptic devices, it will soon become as common as an interaction with mice. Nonetheless, the problems of computer implementations on human vision and touch restrict the integration of the two pipelines. Vision is rather passive process while touch is active and bi-directional. For interactive visualization 30 fps is a sufficient update frequency,

however for haptic applications 1000 Hz is a commonly used update rate. These two rendering pipeline--visual and haptic--compete for CPU and impose serious restrictions on implementations, especially when networked visualization and haptic interactions are considered.

On the other side, the conflict between the large size of commonly used polygon-based models and the limited Internet bandwidth also restricts collaborative shape modeling to practical applications. With the development of computer hardware, the limitation of Internet bandwidth becomes significant, especially for real-time collaborative applications in shared virtual spaces. Although ideally this problem could be solved by increasing Internet bandwidth, it is more practical to solve it by reducing transmitted data size.

A survey on web visualization and shared virtual environments methods and tools leads to the conclusion on the active research niche for development of new data models and frameworks suitable for visual and haptic rendering in shared virtual spaces. The hypothesis of the thesis is that using mathematical functions in virtual object definitions would greatly help to collaboratively model and render them as well as to haptically interact and collaborate in the shared virtual scenes. This hypothesis will be testified and proved throughout the PhD thesis based on its behaviour, coverage and efficiency.

1.2 Objective

This thesis focuses on proposing a practical approach to solve problems of collaborative shape modeling. The approach should support visual and haptic collaboration while providing for faster model exchange and any precision of the representation. This is a

novel issue and it involves many challenging research and development works to be done, which constitute the content of the thesis. Therefore four objectives are defined to achieve the research niches.

The first objective is to propose a novel modeling paradigm to uniformly represent virtual objects' geometry, visual appearance and tangible physical properties. To provide for a faster model exchange and any precision of the representation, mathematical functions and procedures could be adopted for property definitions.

The second objective is to propose an innovative modeling framework in shared virtual spaces, where the visual and haptic pipelines could complement each other to provide a solution to problems requiring collaboration on the web. Mathematical functions and procedures should also be adopted to provide rapid information transmission and flexible interactive modeling operations.

Besides the proposals of the modeling paradigm and collaborative framework, it is also quite essential to implement them with corresponding software tools and core algorithms on several platforms in order to validate their research values. Therefore, the third objective is to design and make comprehensive implementations with different platforms and environments.

The fourth objective is to demonstrate through several examples and practical collaborative applications that the proposed approach could solve problems in current collaborative shape modeling and therefore the usability of the proposed research ideas can be demonstrated.

1.3 Proposed Approach

To achieve the objectives, the following aspects should be taken into consideration:

First, the virtual objects' geometry, visual appearance and tangible physical properties have to be uniformly defined in their own coordinate domains and then merged into virtual objects in the application problem coordinate domain. Geometry samples colours and geometric textures. Physical properties are defined in geometric haptic containers which are not necessarily equal to the surfaces of the virtual objects. Geometry with visual appearance guides haptic interaction in most of the cases however it is not always so. To provide for a faster model exchange and any precision of the representation, mathematical functions and procedures could be used. Hence, implicit functions are good for defining geometric surfaces, explicit functions can be used for defining solid objects, set-theoretic operations, colours and physical properties, and parametric functions are good for defining curves, surfaces, solids and force vectors.

Second, a client-server collaborative framework has to be proposed in order to provide for various collaborative applications. Core concepts of the collaborative framework should also be defined. Therefore virtual objects can be declared as *shared objects* which visual and physical properties are synchronously rendered by all the client computers as they change. *Virtual tools* are shared objects associated with interactive and haptic devices. *Shared events* are used together with shared objects, which are inherited from the event transmission where an event from one client is sent to other clients either through server or directly. No matter which way the platform adopts, *synchronization* is performed to allow multiple users to immerse into the virtual scene and share it. To ensure synchronization among all clients, *consistency control* algorithms such as *locking*

mechanisms are required. Shared objects can be locked by the user for their private use and two locking modes should be used in different situations.

Third, several exemplar implementations have to be developed. The implementations will be built on widely accepted standards, such as Virtual Reality Modeling Language and its successor Extensible 3D. The implementations should allow users to freely build, modify and exchange rich-property models through the Internet. Haptic collision detection and rendering algorithms should be implemented to identify the location of the haptic tool in the scenes as well as to consistently and seamlessly determine physical properties when the haptic tool moves in the scenes with objects having different sizes, locations, and mutual penetrations. Algorithms for information transmission, synchronization and consistency control should also be implemented to provide for reliable and fast interaction among client computers.

Last but not least, based on the implementations, several comprehensive examples and practical collaborative applications have to be developed in order to validate the research ideas and verify their development value. Such examples have to be self explanatory. Comparisons with other methods should also be done to show advantages of the proposed approach. Practical applications should involve real-life input and be able to handle various special cases and situations to show reliability and practicability.

1.4 Thesis Organization

This thesis is organized as follows. A survey of the relevant visual and haptic rendering methods is presented in Chapter 2. In Chapter 3, we review research on visual and haptic collaboration. A uniform approach for visual and haptic modeling and rendering is

described in Chapter 4. In Chapter 5, we illustrate a visual and haptic collaborative framework based on the uniform approach. Implementation details of the software tools and algorithms are demonstrated in Chapter 6. In Chapter 7, we discuss comprehensive examples and practical collaborative applications based on the implementations. Finally in Chapter 8, the conclusion is drawn, and the future work is discussed.

Chapter 2 Visual and haptic rendering

In this chapter, a survey of the relevant visual and haptic rendering methods is given. In Section 2.1 we compare the human vision and touch perceptions. In Section 2.2 we briefly discuss visual modeling and graphics rendering. Haptic modeling and rendering are discussed in detail in Section 2.3. Finally in Section 2.4 the summary is given to conclude the deficiencies of current visual and haptic rendering.

2.1 Vision and Touch

Through vision we receive most of the information about the world around us. This is rather a passive information collection process while touch is an active and bi-directional process. It allows us to perceive tangible properties of objects through stimulation of skin which has different sensitivity in different parts of our body. We also manipulate objects as well as exert forces to receive a force feedback from them.

When implemented with a computer, visual and haptic rendering pipelines originate from the virtual object definitions and end up at the visual and haptic stimuli rendered on the graphics displays and haptic devices. With the graphics displays we “sample” the visual properties of the scene while with the haptic devices we “sample” tangible physical properties of the objects in the scene. Graphics rendering gives us more information as it shows the scene all at once represented by numerous pixels constituting the image, while haptic rendering is mostly a dynamic process where we acquire and accumulate information about objects in the form of a force feedback by moving haptic actuators by object surfaces and inside them. The visual and haptic rendering pipelines are independent, but they are also interrelated and complement each other.

2.2 Visual modeling and graphics rendering

Visual modeling is the procedure of representing 3D models on computers through various approaches. The 3D models describe visual properties of objects, such as geometry, colour, and texture. Although real objects are solid, the 3D models may or may not be solid since object appearances are mainly represented through the exterior. Therefore 3D visual models could be classified as surface models and solid models, which are often represented by using polygons and voxels, respectively. Other representations such as point clouds and NURBS could also be used, but the most commonly used representation is still based on polygons. No matter which representation is used, geometry is always attended first, followed by other visual representations and graphics appearances.

Graphics rendering is the procedure of generating images from computer generated 3D models, which is one of the major topics in computer graphics. Graphics rendering could be classified as pre-rendering and real-time rendering. Pre-rendering is usually used for computationally intensive applications such as movies, in which real-time rendering could not be achieved due to hardware limitation. It could provide high quality rendering results but could not offer good interactivity. On the other side, real-time rendering considers interactivity as the primary goal. Due to the critical fusion frequency, the interactive rendering requires at least 30 frames-per-second update rate. Typical real-time rendering applications include computer games, animations and shape modeling applications.

There are many software tools for visual modeling and graphics rendering. Some tools are based on certain modeling languages, such as VRML, X3D and COLLADA.

These tools are easy to learn and use, therefore they are best suited for fast prototyping and interactive application. They hide certain implementation details to users such as polygon generation, lighting and shading, and leave them to the content viewer. Since certain details are hidden to the users, the rendering result may be not as good as those from tools which provide full user control.

Other software tools are based on programming libraries, such as OpenGL and VTK. They provide detailed control over the modeling procedure, and the users have full control on all implementation details. When combined with advanced rendering algorithms, the rendering results could be much better than those from modeling language based tools. However, the users will have to spend much time to learn the library based tools before they could use them.

Visual modeling and graphics rendering are quite mature research areas in computer graphics while haptic modeling and rendering are still rather exotic. Therefore only related concepts and classifications for visual modeling and graphics rendering are briefly illustrated in this section and we will pay more attention on surveying haptic modeling and rendering in the next section.

2.3 Haptic modeling and rendering

Recent evolutions in computer graphics provide us the ability to not only see, but also to touch the objects in the virtual environment. In [1], the author analyzed the role of touch in enhancing the sense of presence in virtual scenes.

Haptics is a term which refers to the incorporation of touch sense into computer programs by providing force feedback which is delivered by haptic devices. The whole

procedure from the user manipulation to the final device feedback is called haptic interaction.

Until recently, applications about haptic technology are continuously expanding. They are now widely accepted and applied in painting [2, 3], sculpting [4, 5], medical training [6], data visualization [7], rehabilitation [8], virtual mock-up [9, 10], virtual assembly [11, 12] , design review [13, 14, 15] , etc.

There are various kinds of haptic devices available which could be classified by various criteria.

By the functionality of interactions, haptic devices could be classified as Finger-Based devices [16, 17], Hand-based devices [18], Exoesqueletal devices [19, 20, 21, 22], and Tactile devices [23, 24, 25, 26].

Based on the way how human perceive external substances, haptic devices could be classified as tactile devices and force/torque devices. Tactile devices are fitted onto human skin so as to provide sensations that could be perceived by the skin receptors, such as temperature and smoothness. Force/torque devices are usually equipped to human hand and provide sensations that could be perceived by muscles and joints, such as rotation and translation. Throughout this thesis, we are only focused on those haptic devices which are capable of providing force feedback.

By the number of haptic Contact Points (CP) they have, haptic devices can be classified as Single Contact-Point device and Multiple Contact-Point device. Most haptic devices used in research works have only one contact point. They are designed as a stylus attached to a robotic arm allowing for navigating a virtual instrument in a 3D space. Other haptic devices have more than one contact point and work in parallel, such as DataGlove

from Sun Microsystems and CyberGlove from Immersion Company.

Single contact-point haptic devices have both input and output degrees of freedom (DOF). Input DOF indicates how many DOF the user could manipulate the devices with while output DOF indicates how many DOF the haptic devices could feedback to the user. Depending on the targeted research area, there are mainly three categories of haptic devices based on input and output DOF. The simplest and most affordable device category has 3 DOF input and 3 DOF output, which are *x, y, z* input and one 3D force vector output. Devices with 6 DOF input and 3 DOF output provide more flexible user input (*x, y, z, roll, pitch, yaw*) while achieve good balance between the capability and the price. The most capable single contact-point haptic devices are those with 6 DOF input and 6 DOF output, including one 3D force vector and one 3D torque vector.

There are a number of companies producing different haptic devices and SDKs on the market.

Sensable Technologies [27] provides both haptic hardware and development software. The company has several haptic models from the Premium models which provide the largest workspaces and highest 6 DOF force output capabilities to the lowest PHANTOM Omni devices which can offer affordable desktop solutions. Sensable provides two different generations of SDK. The first generation is well known as GHOST SDK [28]. It is a commercially available API that is aimed to offer programmers an easy solution to add haptic force rendering into applications. The second generation of SDK--OpenHaptics toolkit [29]--has a different structure from GHOST SDK. It provides a more comprehensive architecture that contains two layers, which have different capabilities. The foundation layer is the Haptic Device API (HDAPI), which enables the

users to render forces directly and control low-level runtime details through the driver. Haptic Library API (HLAPI), on the other hand, is a higher and simpler layer that is built on top of HDAPI and resorts to OpenGL for graphical rendering. It is mainly designed for fast incorporation of haptic interaction into existing OpenGL programs, which allows significant reuse of existing OpenGL code and greatly simplifies synchronization of the visual and haptic threads.

Immersion company [30] provides different kinds of haptic hardware and SDKs in several application areas. VibeTonz Studio SDK provides the ability of integrating haptic sensations into the mobile phone-based ringers, alerts and games. Immersion AccuTouch medical simulator is aimed at providing realistic and safe haptic experience for surgical and medical simulation systems. Immersion also has a 3D Interaction product line that produces hand-centric hardware and software (VirtualHand SDK) solutions for animating hand movements and allowing users to manipulate graphical objects with their hands.

Force Dimension company [31] offers various haptic devices. Its omega.x architecture is able to provide a high-precision force feedback with different levels of performance and modularity that can be suitable for different applications. Force Dimension also provides a low-level software API, which offers the ability to render customized haptic effects on their haptic devices. For high-level development, Force Dimension supports the CHAI 3D [32] framework from Stanford University.

Quanser company [33] produces 3 DOF and 5 DOF haptic devices. Its planar twin-pantograph structured haptic device has three degrees of freedom, which allows for planar translation and unlimited rotation about a single axis. Its 5 DOF haptic wand allows for three translations and two rotations (roll and pitch). The company also provides

Q8 SDK along with the device driver that helps users to develop their own applications.

MPB Technologies company [34] produces several different haptic devices. The Cubic-3 is a 3 DOF haptic device while the Freedom 6S is a high fidelity force feedback device operating in 6 DOF. The Freedom 7S is built on top of the Freedom 6S. It offers the advantages of full 6 DOF with an interchangeable force feedback scissors grip, which is ideal for medical simulation and master/slave robotics where sensitive control by a scissors-like handle is required.

ERGOS Technologies [35] provides several high-fidelity haptic interfaces with different DOFs. Its sliced motor allows the user to add additional slices to increase the number of DOF of the base. It also provides low-level control to the devices so that the users could access to the analogue setup directly. Besides, it allows the user to choose specific morphology according to the user's need such as keys, grasps, bows, 3D/6D sticks and new-designed end-effectors.

Novint [36] produces Novint Falcon as a low cost and high fidelity haptic device that is mainly for consumer computing. The low-cost Falcon is able to help the user haptically feel the virtual environment when they are playing haptic enabled games. Besides, Novint offers a free SDK [37] to support various haptic applications. Although the SDK is not as sophisticated as Open Haptic Toolkit from Sensable, it would still be a great help to introduce haptic devices and applications to mainstream markets.

Typical haptic interaction concentrates on two issues: Haptic Collision Detection and Haptic Rendering. By considering time in discrete fashion, Haptic Collision Detection constantly checks if the virtual representation of haptic contact point collides with a certain substance in the virtual scene. The virtual representation of a contract point can be

a point, a vector frame of reference, or a 3D object. Haptic Rendering refers to generating the force feedback to the user depending on the position of the contact point with reference to the objects in the haptic scene. It conveys to the user additional information about the virtual environment synchronously with its visual rendering.

2.3.1 Haptic Collision Detection

Collision detection is a general challenge and fundamental problem in various research areas such as physically-based modeling, molecular modeling, computer animation and robotics. Haptic collision detection is similar but not identical to traditional collision detection in computer graphics. Besides the update rate difference shown in Figure 2.1, they also differ in the way how the virtual objects in the scene were defined. Prosperity of haptic research accelerated the demand for better collision detection algorithms. However, most collision detection approaches have some restrictions or assumptions on the objects to be checked, and quite frequently those solutions are not universal.

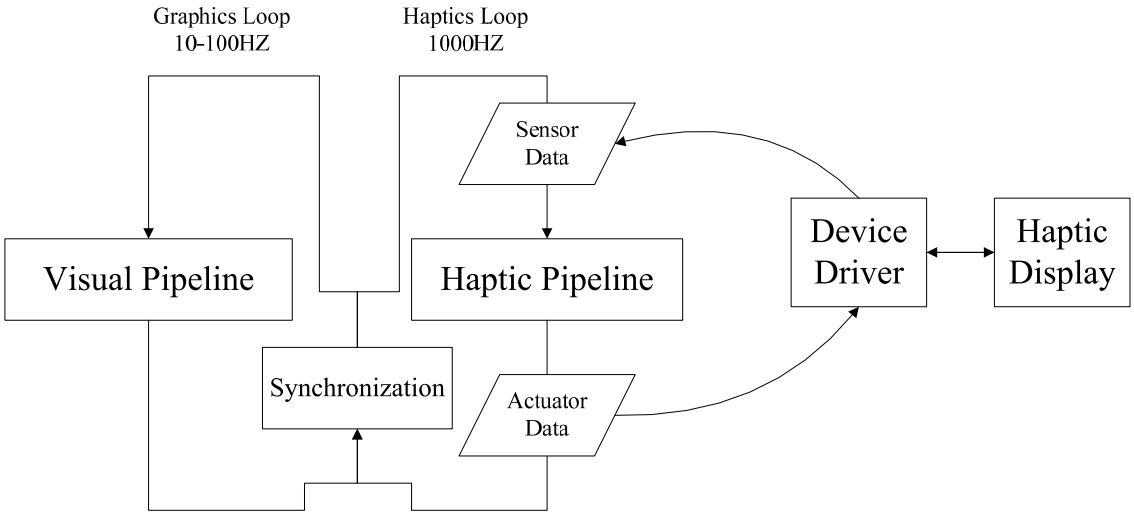


Figure 2.1 Architecture of general visual and haptic pipeline integration

Although there have been efforts in performing collision detection with curved solid

models [38], high density points [39], polyhedral models [40], superquadric models [41], subdivision surfaces [42], and implicit functions [43], the most commonly used approach is still based on collisions with polygon meshes. There have been several popular haptic collision detection algorithms and software available. In [44], Ruspini proposed the idea of using a hierarchical bounding sphere for haptic collision detection. GAMMA (Geometric Algorithms for Modeling, Motion, and Animation) Research Group from University of North Carolina designed and implemented H-Collide [45], which adopts spatial subdivision and OBB trees for haptic collision detection.

The basic idea of collision detection is to make fully pair wise intersection test for all objects in the virtual environment. However, this procedure would be time consuming in real life scenarios and interaction could not be achieved. A feasible solution to this problem is to reduce the number of exact collision detections and use hierarchical scheme for multi-level collision detections.

Another way to improve the interactivity of Collision Detection is to adopt frame-to-frame coherence techniques. The principle of frame-to-frame coherence is that a scenario does not have obvious changes between two successive frames. If time is considered as discrete time episodes such as frames, there will be a little transformation of objects from frame to frame, thus the general collision detection situation will not have big changes and certain detections could be avoided to increase the interactive speed.

Besides, physics engines also support for fast collision detection, which provide possibilities to haptic collision detection. In [46] the authors incorporated Physx [47] and CUDA [48] to accelerate haptic collision detection and simulation by building an intermediate layer between haptic SDK and Physics Engine SDK. Other physics engines

such as Havok Physics [49], Open Dynamic Engines [50], Bullet [51] and Newton Game Dynamics [52] could also provide similar accelerations.

Nevertheless, collisions with polygons suffer from a problem that they rely on a primitive level input such as vertices. For standalone applications, collecting primitives is fairly easy, however, for other applications, such as plug-ins to existing software tools, the primitive level collision detection algorithms are not feasible since the existing software may not provide APIs for retrieving primitives. This issue can be solved by using implicit functions. For implicit functions, it is rather trivial to implement the collision with the contact point membership predicate. In [53], Sourin and Wei proposed to haptically collide with implicit functions without a need of the primitive level input. In [43], another implicit function based approach was proposed for rendering large geometry models at 1000 Hz rate however it still requires surface information retrieval from volumetric data for force generation.

2.3.2 Haptic rendering

Haptic rendering mainly concerns two issues: position/orientation of the contact point and contact force that is sent back to the user. Naturally, as well as historically based on the existing applications, haptic rendering follows visual rendering pipeline. This influenced the most common way of haptic rendering with the visible surfaces which are typically displayed using polygons. Various ways can be applied to calculate forces with polygonal models however the most popular two approaches are Penalty-based approach and Constraint-based approach.

Penalty-based approach directly maps the position and orientation of the haptic

contact point into the haptic scene. Collision detection is purely based on the real haptic contact point, and forces are generated when the collision has happened and the haptic point has already penetrated objects.

Constraint-based approach defines a proxy of the real haptic contact point. Typically, a proxy would be a point, although it can also be a sphere, a crosshair or other shapes. When the proxy is a point, it is also referred to as the SCP (surface contact point), as shown in Figure 2.2 (taken from the Open Haptic Toolkit manual). A proxy follows the movement of the haptic contact point, but it is constrained on the object surface. When the proxy collides with the objects in the virtual environment, a force vector will be calculated based on the minimum energy configuration between the contacted surface and the haptic contact point.

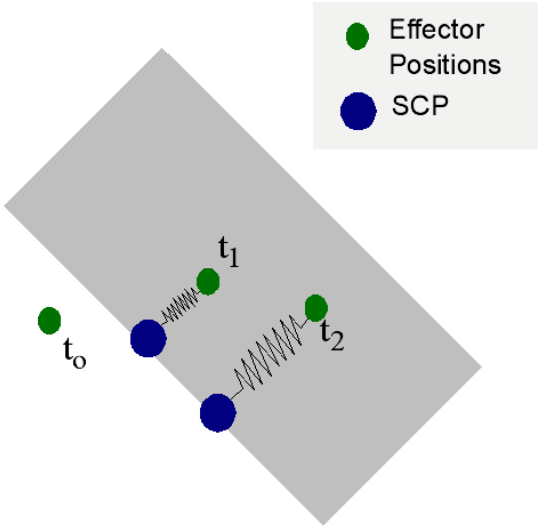


Figure 2.2 SCP and spring force model

In [54], Zilles and Salisbury first introduced a name of the haptic proxy as “god object” and then extended the “god object” idea and proposed approaches to implement force shading, surface stiffness, friction, and haptic texture.

Although penalty-based and constraint-based approaches are popular for haptic

rendering, they suffer from the same problems as the polygon-based visualization–polygon complexity. After a certain number of polygons participated in the scene, haptic rendering cannot be performed with the required 1000 Hz frequency. Though it is possible to continue with interactive rates even with a frequency as low as 300-600 Hz [55, 56], it still imposes at least an order of magnitude higher demand to the processing speed. As a result, compared to the maximum number of polygons that can provide for smooth visual interaction, the typical number of polygons providing for smooth haptic interaction is very much below it. Based on our laboratory test on a mid-level workstation, by using general polygon rendering approach and standard haptic interaction SDK, the fast update rate for smooth haptic rendering can only be achieved with no more than one million polygons. In comparison, the same workstation is able to smoothly render millions of polygons visually.

There are efforts in solving this problem from several perspectives, such as hierarchies of bounding volumes [57], spatial partitioning [58], and GPU-based acceleration [59]. Other approaches have also been studied, such as a velocity driven haptic rendering approach [60] which is based on LOD and render either coarse or detailed models depending on the velocity the user moves the haptic point.

Furthermore, things get further complicated if haptic rendering is performed not only with an elementary haptic object but with a more complex geometric object representing the virtual haptic tool. Collision detection between objects with or without physics will reduce the performance in terms of the participating polygons even further. In [61], the authors proposed a combination approach for accelerating object-object contact detection, such as geometric locality, temporal coherence and prediction.

Besides haptic rendering with the visual surfaces, there are also a few works on haptic rendering of the interiors of solid objects and forces that can be directly used for calculation of force feedback. In [62], a voxel sampling approach has been proposed to accurately render complex real-world task. In [63], a solution was proposed to haptically render the volume data using dynamic spline-based implicit functions. In [64] Kim et al. proposed a method of touching a virtual volumetric non-rigid object with the object at multiple points. In [65, 66] Lundin et al. proposed proxy-based haptic interaction with volume MRI data based on domain knowledge filtering to extract separate material and surface information and to avoid explicit classification. The result has been used on SenseGraphics (Volume Haptic Toolkit) VHTK [67] which is a volume haptic toolkit that lets one explore scientific and medical data sets using haptics and 3D visualization. VHTK comes with functionality for scalar and vector visualization methods and includes haptic modes for surface, gradient and viscosity force rendering.

2.3.3 Current state-of-the-art of haptic SDKs and applications

Currently there are several haptic SDKs available for building a complete haptic interaction pipeline: CHAI3D, Reachin API&HaptX, OpenHaptics toolkit, Novint SDK and HAPI.

CHAI3D is an open source SDK that supports different haptic devices from different companies. It adopts OpenGL for visualization, ODE (Open Dynamics Engine) for rigid body collision detection, and GEL Dynamics Engine for simulating deformable objects.

Reachin API utilizes C++, Python and VRML to provide users with a flexibility of implementing haptic programs. HaptX is a haptics engine from Reachin, which works as

a haptic toolbox for easy and fast development. It also works with different physics engines for interaction.

Open Haptic Toolkit from Sensable encapsulates functionalities for High-Level Collision Detection (HL API), which provides useful functions to directly handle geometric primitives and transformations. On the other hand, it constrained the scope of usage (only useful when the developer knows the polygons). The low-level HD API could provide better flexibility to the developers however they have to design their own collision detection algorithm. Besides, even though the Open Haptic Toolkit is very good, it only supports Sensable devices—other vendors and users have to develop their own SDKs.

Novint SDK is similar to HDAPI in Open Haptic Toolkit however it is simpler and provides less API functions.

HAPI is developed by SenseGraphics and is intended to be a haptic rendering engine. It is device independent and purely built with C++. It utilizes OpenGL/DirectX for graphical rendering and OpenHaptics toolkit/CHAI3D/God Object/Ruspini for haptic collision detection and force rendering.

However, these haptic SDKs and applications have several drawbacks.

First, various haptic sensations such as feeling of surface tension and friction, as well as solid object viscosity and force fields, can only be provided separately and often at predefined, known for the haptic algorithm, locations of the haptic tools, while concurrent haptic rendering of these phenomena at any random location of the haptic tool is still a challenge. As an example of such efforts, CHAI3D supports multiple material effects such as viscosity, vibration and stiffness. However they implement stiffness as elasticity

which prevents from penetrating the surface of the objects to enter the interior part to feel the viscosity.

Second, although several haptic effects have been identified and proposed for haptic rendering, the ways how to define a believable haptic rendering using these effects are yet to be studied. Many haptic research works aim at more precise force calculation and rendering, while how the forces could be defined to provide the best immersion is less discussed. For example, in CHAI3D, each haptic effect has only a few parameters exposed to the programmer, which are individually hardcoded in the C++ source file. Each haptic effect can only be rigidly defined and applied onto the whole object, and there is no way to render an object with complex and flexible haptic effects, such as a variable stiffness across the object surface and a variable viscosity inside the object. This limitation greatly affects the content creators to simulate real-life objects using haptics.

Third, usually only one type of object can be found in haptic scenes such as polygon-based surfaces, function-based surface models, or volumetric-based solid models. This simplifies the haptic interaction algorithms but greatly limits the abilities of the content creators.

Last but not least, commonly haptic scenes are simplified and tuned to perfect demonstrations. For example, the sizes of the haptic objects in a scene are always of the same order of magnitude so that the case of concurrent haptic rendering of very large and very small objects is simply excluded. Also, the haptic workspace is usually defined with a rigid size and location without abilities of scaling it up or down or moving it forward or backward for reaching objects with different sizes and locations. The initial position of the haptic contact point is usually placed outside the objects in the scene, which excludes

the situation of being initially inside objects with viscosity or force field defined. Besides, haptic SDKs seldom consider inclusions or intersections of objects with different haptic properties. For example, in CHAI3D, when a small sphere is located completely inside a big sphere, properties from both spheres will be rendered when the handle is inside the small sphere, which may be explained as reasonable but could be unrealistic for certain applications, where only properties from the small sphere need to be rendered.

2.4 Summary

In this chapter a survey of visual and haptic modeling and rendering is presented. Concepts and classifications of visual modeling and graphics rendering, as well as haptic modeling and rendering are demonstrated. A detailed review of haptic collision detection, haptic rendering and current state-of-the-art of haptic SDKs and applications are also given. By comparing the way how humans perceive vision and touch, as well as how computers implement visual and haptic pipeline, the problem is elicited that the two pipelines are heavily competing for CPU resources and therefore generate serious problems in efficiently and uniformly render both visual and physical properties defined for virtual objects. Therefore, we can conclude that novel efficient models are in urgent need to solve the above-mentioned problems.

Chapter 3 Visual and haptic collaboration

In this chapter, a comprehensive survey of visual and haptic collaboration is given. In Section 3.1 we discuss networked visualization and collaboration. Networked haptic interaction and collaboration is demonstrated in Section 3.2. Finally, in Section 3.3 the summary is given and the problems of current visual and haptic collaborations are elicited.

3.1 Networked visualization and collaboration

The advent of Internet has opened a way for communication, and the World Wide Web has become the most efficient and convenient platform for distributed access to information and collaboration. The increasing demand for high-performance and large scale projects is a main impetus for the collaborative visualization on the web. As the computer networks improve, high bandwidth and lower latency connections are widely adopted. The current widespread deployment of network hardware and software technologies has created an opportunity and need for new collaborative methods and techniques in the creation of middle-ware infrastructure to support sophisticated collaboration over wide area networks.

One obvious advantage of networked visualization and collaboration is that it allows geographically separated users to take advantage of a larger number of resources located at different locations for investigation and cooperation. Such applications are usually large and crucial, which require strict stability and performance. In [68], Nigel W. John et al. introduced the web-based surgical educational tools in the WebSET (Web-based

Standard Educational Tools) project, which was implemented based on XML, VRML, JaCoB physiology simulator and Deep Matrix. The system is able to provide both multi-user and single user training procedures with good immersion and support for various multimedia content.

In [69], Gerhard Reitmayr et al. addressed three issues for a typical design and implementation of networked visualization and collaboration.

1. Data sharing

To build a networked platform for visualization and collaboration, data has to be shared among different clients. Depending on the architecture the platform, client-server mode or Peer-to-Peer mode may be adopted for sharing information.

2. 3D and multimedia rendering

A networked visualization and collaboration platform should be able to provide the users immersive experiences through 3D and multimedia rendering. Proper multimedia rendering modules and pipelines have to be the essential part and guarantee smooth visualization.

3. Interaction

For networked visualization and collaboration, the users should be able to interact with each other as well as with the objects defined in the virtual environment.

3.1.1 Classification of approaches for networked data transmission

In a collaborative environment, there are several approaches available for data transmission: broadcast transmission, unicast transmission, and multicast transmission.

Broadcast transmission refers to one-to-many communication between a sender and

the rest of peers in the collaborative environment. Such transmission has no configuration to block transmitted data to certain receivers; therefore it is very inconvenient to adopt broadcast transmission in a collaborative environment.

Unicast transmission refers to one-to-one communication between a sender and a specified receiver. It is mostly adopted in LAN configurations such as HTTP and FTP. Unicast usually employs the TCP protocol to ensure reliability, and each active communication will require additional bandwidth. Therefore when multiple communications are running concurrently, the consumed bandwidth would soon reach its overhead. This drawback seriously affects its use in collaboration.

Multicast transmission refers to one-to-many communication between a sender and zero-to-many specified receivers. Multicast transmission does not require information of the receivers, such as how many receivers are there and who they are. The sender just sends out one copy of transmitted data once, and the receivers will be responsible to identify and acquire necessary transmitted data. Since only one copy of the transmitted data is sent, multicast transmission is quite bandwidth-saving and efficient, especially when the number of receivers is large. Multicast usually employs the UDP protocol to ensure efficiency; however since UDP is not reliable, certain error detection and retransmission mechanism are required to ensure correct transmission.

By analyzing and comparing the advantages and disadvantages of the three transmission approaches, it is obvious that multicast transmission is best suited to be adopted for collaborative environment.

3.1.2 Classification of networked visualization and collaboration

Networked visualization and collaboration can be achieved in several ways based on the multicast approach. By the way they transmit data we can classify them into three categories

1. Image transmission

Images are the final output of visualization which could be directly perceived by the users. Therefore they are the most intuitive and straightforward media for data transmission in collaborative environments. For image transmission, all visualization procedures are solely done by servers, and the data transmission occurs only after the server has accomplished the computations. Image transmission is usually built on thin client – strong server organizations. Grid rendering could also be adopted if real-time output of high quality images cannot be achieved on single server. There are two commonly used approaches for grid rendering, the first approach renders images in parallel as each frame of the image sequence is independent, while the second approach splits each image into tiles and distributes tiles to each grid node for rendering.

By employing the powerful hardware of servers, it is possible to use computationally expensive algorithms to generate the best Level-of-Detail results. Besides, since the server part is responsible for most of the work, various thin clients with low resolution and weak processors could be used for receiving the rendering results at interactive rate. One typical image transmission-based application is OnLive [70], which adopts cloud computing on the server side to run computer games at top rendering quality, sends the rendering result to end-users and responds to the user actions at interactive rate. The users will not need high-end PCs and they could play the latest games on TV, netbook

computer and even mobile devices like iPad [71].

However, image transmission has several problems. The most essential problem is the huge data size, which is very difficult to be transmitted at interactive rate on WAN or Internet. For example, the OnLive platform claims to only support the users living less than 1000 miles from their data enter. The platform also requires at least 1.5 Mbps bandwidth for standard quality rendering and 5 Mbps for HD quality rendering. Even with such high bandwidth, data compression algorithms are still necessary. For example in [72], Anthony Chong et al. proposed a framework for online grid rendering, together with a lossless 3D compression algorithm. Such efforts improved the interaction for image transmission but they still could not totally solve the problem.

Another problem for image transmission is that such architecture is highly dependent on the server. If reliability of the server could not be guaranteed, jitter, delay and lag will happen on all client computers and dramatically decrease the system performance.

Based on the analysis above, a conclusion can be drawn that image transmission is not suitable for deploying interactive networked visualization and collaboration.

2. Event transmission

With the rapid development of personal computer hardware the computational power of personal computers has dramatically increased in recent years. Nowadays, even mid-level personal computers are equipped with very powerful CPU and GPU, which allows for running complex algorithms and collaborative visualization applications. Considering this fact, it is possible to impose the main part of the collaborative visualization platform on the client and consider the server as a centre for transmitting messages and commands. Such architecture in which only events are transmitted is called

event transmission. All other resources are preloaded from the local hard drive of the client. Since the event messages are small in size, event transmission works well even with very narrow bandwidth.

Since the main part of the networked visualization and collaboration is done on the client part, the users will be able to get local responses at interactive rate even for very complicated interactions. This feature is ideal for applications with many concurrent clients and frequent data transmissions, such as online MMORPG (massively multiplayer online role-playing) games which consist of a huge client part and a light-weight server part. When a user starts a MMORPG game, the client will first establish a connection to the server and load all pre-installed models from a local hard drive. When playing the game, what the user sends to and receives from the server are just events of controlling commands. All graphics rendering and complicated calculations such as PK (Player Killing) and battles in the game are computed on the client side. This architecture can dramatically reduce the burden to the game server and therefore the server could support much more concurrent clients compared with other architectures. For example, it is reported in 2009 that the number of concurrent online players for World of Warcraft—the most successful MMORPG game—had already exceeded one million.

Nevertheless, the drawbacks for event transmission could not be neglected. Since the computation power of the client is still incomparable with the server part, advanced rendering algorithms such as ray tracing and fast polygonization could not be adopted on the client side. Although the client hardware and rendering algorithms are improving fast, the rendering quality is still incomparable with those produced by image transmission.

Besides, since different event transmission applications will require pre-installation

of different models and rendering tools, the client computer will have to maintain all of them, which is quite troublesome. Things get further complicated when such applications require updates of certain models and tools. In that case, all client computers across the world will have to download and install the updates. Based on the analysis above, it is clear that event transmission is not suitable for rapid network publishing.

3. Model transmission

Besides image transmission and event transmission, model transmission is also a commonly used approach for data transmission. Model transmission neither overburdens the server nor requires huge pre-installed data on the client part. On the contrary, it logically separates networked visualization and collaboration into two parts, with rendering procedure performed on the client part and models sent from the server part. In model transmission, the difference between the client and the server is not quite obvious therefore either client-server mode or peer-to-peer mode could be adopted. For client-server architecture, tasks closely related to the pre-processing phase, such as raw data interpretation and model generation, can be put on the server side, while other tasks, which require immediate response and real-time visualization, can be put on the client side. For peer-to-peer architecture, each peer will first generate the models to be transmitted and then exchange. The receivers are responsible for rendering the local results using the acquired models. Render quality is controlled by individual clients therefore the users could choose from different settings to best suit the condition of their network and local hardware.

Since only models are transmitted between networked computers, model transmission requires relatively narrow bandwidth and small client software. The small

client could be easily downloaded and updated. This makes model transmission the most suitable approach for networked visualization and collaboration, especially when both rendering quality and interactivity are required.

However, the disadvantages for model transmission could not be ignored. When the transmitted models are large, the model downloading procedure will take a long time before the users could visualize and manipulate with the results. This problem could be solved by adopting model transmission acceleration approaches, such as progressive delivery [73, 74] and model compression [75, 76]. Besides, in [77], a view-oriented approach was proposed to efficiently set priorities for different views of the data and therefore provide faster model transmission and high-quality QoS management.

3.1.3 Model transmission-based networked visualization and collaboration implementations

Since we consider model transmission the most suitable approach for networked visualization and collaboration, more details about model transmission will be discussed in this subsection. Therefore by the relationship of contents and owners in implementing networked visualizations and collaboration, two approaches can be classified.

Some model transmission based applications implement the language itself as the platform. In this case, the content stays with the owners who can freely move from one provider to another. Examples which belong to this category are VRML [78], X3D [79] and COLLADA[80].

VRML (Virtual Reality Modeling Language) is a standard file format for representing 3D interactive graphics. In 1997 VRML was proposed and accepted as an

International Standard. VRML files are generally hierarchical collections of objects, which consist of polygon-based geometries, appearances, transformations, viewpoints, etc. It also provides interactivity by using event scripts and routing mechanism.

X3D (Extensible 3D) is the successor to VRML. It introduces plenty of new features to enhance VRML while still provides backward compatibility with it. X3D offers enhanced APIs for better expandability as well as several encoding mechanisms including XML encoding, binary encoding and VRML encoding. Another major feature of X3D is the ability to provide extensibility. Four baseline profiles are proposed in order to best suit for specific application requirements: Interchange, Interactive, Immersive and Full. Each Profile contains certain components, Interchange is the basic profile which provides maximum portability and compatibility while Full is the complete profile which provides maximum functionality and application potential. Based on the scope of this PhD project, we adopt the Immersive profile throughout this thesis.

COLLADA (COLLAborative Design Activity) is a file format for 3D interactions. It is mainly designed as a container for 3D data and mostly used as an exchange format for 3D authoring tools. There are several software tools developed using COLLADA, such as Unreal game engine and Google Earth. COLLADA adopts XML database schema to enable 3D authoring applications freely exchange digital assets without loss of information.

Other model transmission based applications implement the viewer, especially the Internet browser as the platform. In this case, the content availability depends on the server part, and owners cannot freely move from one platform to another. Both client-server and peer-to-peer organizations can be adopted for such applications.

Examples of this category are Open Croquet [81] and Open Cobalt [82] with squeak language, Second Life [83] with Linden Scripting Language (LSL), Active Worlds [84] with RenderWare scripts, etc.

Open Croquet is an open source software development environment for creating and deploying collaborative multi-user online applications on various operating systems and devices. Open Cobalt is a current continuation of Open Croquet which is built on top of Croquet SDK. Both Open Croquet and Open Cobalt use Squeak language for programming and feature for a peer-to-peer based network architecture, which supports real time collaborative applications and cross-platform 3D applications such as collaborative scientific modeling and design visualization. The squeak language is a cross platform language which is purely object-oriented and editable while running. Therefore it is very efficient in real-time interactions.

Second Life is a 3D virtual world in which players have the freedom to build objects and interact with each other. It is based on the client-server organization and supported by a grid of two thousand servers [85]. The client is mainly responsible for visualization and local interaction while the server part is responsible for monitoring and interacting with the client part. Each server only takes care of a certain area of the virtual environment, therefore if one user wanders from on area to another, the servers will automatically switch the control correspondingly. Besides, the server part is also in charge of collision detection, storing object state and calculating visibility on objects. Second Life provides two ways of building objects: using geometric primitives and using a scripting language. The scripting language is named LSL (Linden Scripting Language), which is a C-style language that enables users to define and control the behaviour of objects. The script

defines both the behaviour and property of objects, thus providing rich appearance and interaction in the virtual world.

Active Worlds is a browser-integrated 3D virtual reality platform that can run on several operating systems. The virtual world is similar to Second life, which provides users the freedom to explore 3D virtual environments as well as build their own 3D contents. The integrated browser acts as an instant messenger which can handle voice chats and web browsing, as well as building structures and areas from selected objects. Active Worlds supports objects stored as RenderWare script (RWX), which defines vertices, polygons and material information of virtual objects as well as supports for collision detection, lighting, reflection and shadows.

There are also quite a few research works on model transmission based network visualization and collaboration. In [86], Bajaj and Cutchin proposed a model which defines the computation loop called DSAV loop (Data sources, Simulation servers, Analysis tools and Visualization clients) that supports interactive user control of distributed simulations. Based on this model, they implemented a web-based multi-user interface using Java, CORBA and VRML. In [87] Konduri and Chandrakasan presented a collaborative and distributed Web-based CAD framework that enables designers collaborate on a design and efficiently utilize existing design tools over the Internet. In [88], Chastine et al. presented a low-cost collaborative virtual environment for molecular modeling and visualization. In [89], Okada et al. illustrated that collaboration on the web can also be applied to environmental education (EE). They adopt the client-server architecture and utilize Java and VRML for platform implementation. In [90], Halvorsrud et al. introduced the Mission Queen Maud Land (MQML)--a collaborative virtual

environment that aims to stimulate secondary school pupils collaborating in grasping complex subject matter. They adopt the client-server architecture and use C++ and VRML to describe the platform implementation. In [91], Tang et al. introduced a collaborative feature-based modeling framework, which does not adopt the commonly used token-passing approach and allows for non-locked interactions from different clients concurrently. VRML and Java3D are used for model description. In [92], Alma Martínez et al. proposed a Peer-to-Peer based real-time networked visualization system on a gird of heterogeneous computers. The system adopts dynamic time slots to group transmitted nodes into various areas and therefore ensures minimum data transmission. Time stamps are also used to ensure the correct order of action execution. In [93], Mironova et al. proposed a framework of collaborative volume visualization, which allows several users to collaboratively view and explore .3DS file based simulation visualization over the network. The framework is based on client-server structure and use Java and VTK as the development tools. In [94], Bourne et al. defined a Molecular Scene Description Language (MSDL), which supports interaction, scene retaining and data interchange. It also adopts VRML to visualize the scene, Java to build the console, and Virtual Reality Behaviour System (VRBS) protocol to communicate among clients. In [95], Wang et al. proposed a client-server architecture to collaboratively visualize volume datasets across the network based on VRML. To provide real-time interaction, the authors also proposed a method to pre-process the volume dataset and construct search index.

3.2 Networked haptic interaction and collaboration

With the growth of the Internet population, demands of networked haptic interaction and

collaboration are continuously increasing, which impede great cooperation between haptic interaction and computer networks.

In section 3.1.2, we classified networked visualization and collaboration into three categories based on the way how they transmit data: image transmission, event transmission and model transmission. For networked haptic collaboration, only non-visual data are transmitted. Therefore only event transmission and model transmission remain meaningful.

Event transmission has minimal transmitted data size and therefore it is ideal for fast data transmission in networked haptic collaboration. However, event transmission requires all data to be pre-loaded on all client computers, which is only suitable for interactions and collaborations with existing rigid contents such as networked haptic collaborative games [96]. Applications such as dynamic shape modelling and collaboration are not best suited with event transmission.

Compared with event transmission, model transmission has larger transmitted data size, which is an obvious disadvantage for networked haptic collaboration. However, such disadvantage could be overcome by introducing compact model representations such as F-Rep [97]. Besides, model transmission could represent complex collaborative operations such as editing certain properties of a model, while event transmission is unable to do so.

Based on the discussion above, we consider model transmission also the most appropriate approach for networked haptic collaboration, especially when dynamic content creations and modifications are essential.

We therefore identify three approaches to what could be called networked haptic

interaction and collaboration.

3.2.1 Stand-alone haptic application capable of working with 3D web data

This approach assumes making stand-alone haptic applications that are capable of working with 3D web data. These applications are initially developed for a single computer. After that, they are extended to work with remote 3D data on the web. This approach is quite straightforward yet very primitive, which could not be called “fully network haptic interaction”.

H3D API [98] , a scene graph based API by SenseGraphics Company, can use both C++ and Python to 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. However, this approach does not really lead to creating web-based applications but rather allows for using them alongside with the web.

In [99] 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. This file reader is capable of handling the basic elements of the VRML 2.0 specification, such as primitive shapes, meshes, transform nodes, into a scene graph for a haptic scene. The obtained scene graph is displayed haptically using the GHOST SDK and graphically using the OpenGL libraries. However, this approach has some drawbacks. First, this file reader does not support more complicated VRML constructs such as definitions, ROUTES, extrusions, and appearance node. Second, all the objects are loaded from the source file before the initialization of the plug-in, which means that it cannot deal with

interactive scenes. Asano et al. [100] presented a survey of exhibition planners and visitors about the distributed haptic museum through the Internet, which is featured as distributed and touchable. They use GHOST SDK for calculating the reaction force applied to the user, and use models which have information about the reaction force to be calculated.

McLaughlin et al. [101] built a "Haptic Museum" through the Internet and other network to allow museum visitors to explore 3D works of art by haptically appreciating them, which is usually not possible in ordinary museums. They use the 3Scan system to generate fully-textured models in 3DS format compatible with the rendering systems.

3.2.2 Plug-in to web browser

Making plug-ins to web browsers is another way to implement networked haptic applications. Compared with the first approach, these applications do not have executable programs to retrieve and process remote data. Instead, they adopt just an ActiveX control or a dynamic linking library file called by the web browser. It naturally adopts the "networked" ability from web browser and requires installation of a small plug-in only once, which is quite easy for the users.

The open source CHAI3D [32] is a set of C++ libraries for haptic interaction and visualization. It provides Integration with ActiveX for web-embedded haptic applications. It is device independent and provides several ways to implement the haptic loop. However, each scene that is intended to be embedded into the web browser has its own specific code. The users have to learn how to program from scratch before they are able to implement their own scenes. Also, ActiveX control is not a standard component in W3C

standard and works only in Microsoft Windows, which implies that it may not work on some popular web browsers other than Microsoft Internet Explorer, such as Mozilla Firefox and Google Chrome. Furthermore, even in Microsoft Windows, users will come across security warning each time when they download the ActiveX control, which is really annoying and inconvenient.

In [8] a web-based robotic rehabilitation named “Java Therapy” is described, in which force feedback is incorporated into a Web page using a Java Applet. The force feedback is then used for transferring desired information. Immersion Corporation’s FeeltheWEB ActiveX browser control is also used in this project to manage the force-feedback joystick and receive function called through HTML. Nevertheless, this project still requires users to refine and validate the library of therapy and build new specific scenes.

In [102] [103], HapticWeb is introduced with XVR [104]. HapticWeb is built on CHAI3D, which exposes most of CHAI3D’s features to the scripting system and then further extends them for haptic feedback realism and expressiveness. It adopts “.aam” file format as the model format and .s3d as the scene format, which are compiled and deployed to an HTML web page. However, the user has to learn the C-like XVR Script first, and then compile and manually embed the output binary file into the web browser. Besides, the users have to recompile and redeploy the file for each modification.

3.2.3 Haptic collaborative applications established directly by TCP/UDP protocol

In this approach, haptic collaborative applications are built so that the connection between

the collaborating parties is established directly by TCP/UDP protocols. The architecture of such applications is based on client-server and peer-to-peer configurations as well as their mixtures. Networked interaction and haptic rendering are naturally supported from the very beginning and therefore some optimization on network transmission can be made to ensure the fluency of the 1000HZ dataflow rate. General problems such as jitter, lag, jerky, data loss and disorder of successive data sequences can also be reduced from the low-level part of the platform.

Though quite efficient, this approach may often lack general reusability due to the specific data formats, protocol wrappers and network connections used.

There are quite a few examples of such implementations. Hamam et al. [105] developed a cataract eye surgery simulation using TCP and UDP for synchronizing an access to the scene for different remote clients and with all the graphics models converted to VRML and/or 3DS data formats. The Reachin API is adopted to load the eye model and tools. Shen et al. [106] introduced the C-HAVE—the collaborative haptic, audio, visual environments in a network context adopting VRML-enabled and Java 3D-based browser for graphics rendering and using UDP as the communication protocol based on their Data Distribution Management module. Iglesias et al. [12] described an assembly simulation application in a collaborative haptic virtual environment, in which the users can simultaneously interact within the same scene using traditional or haptic devices. The application is implemented based on client-server architecture and adopts DATum -- an object oriented variational non-manifold geometric modeller as the data format. Basdogan et al. [107] designed an experiment where the users at different locations were asked to jointly move a ring along a wire without touching it through network. They built a simple

but expandable user interface using Open Inventor graphics tool kit and C++ to display 3D objects in networked virtual environments. The interface enables the user to load polyhedral virtual objects from a simple user-defined text file for constructing a multimodal virtual environment that has visual and haptic components. Dan Morris et al. [108] introduced Haptic Battle Pong-- a competitive networked OpenGL game that makes extensive use of 3 DOF and 6 DOF haptic devices for networked haptic interaction. They also proposed a "private haptic space" assumption to solve the synchronization problem. Goncharenko et al. [109] presented distributed PHANTOM-based systems for collaborative haptic visualization of a VR crank model. Fukuda and Matsumoto [110] designed a hybrid architecture for shared haptic virtual environments with a server to handle connection, disconnection and synchronization. The clients communicate with each other using peer-to-peer architecture. Hikichi et al. [111] made a research on the client-server architecture for tele-haptics. Alhalabi and Horiguchi [112] implemented a cooperative shared haptic virtual environment called the "Tele-Handshake", where users can interact and feel each other haptically using client-server configuration.

3.2.4 Problems of networked haptic interaction and collaboration

Since haptic interaction requires faster information updates (1000 HZ) than typical network-based applications, general technical issues about networked haptic collaboration such as jitter, lag, jerky, data loss and disorder of successive data sequences are more frequent to appear and therefore more sophisticated solutions are required.

Zhou et al. [113] proposed two main problems about collaborative haptic applications: how to keep the coherency of virtual scenes among all users and how to get

stable force feedback when haptic information is sent through non-dedicated channels where there is some latency and jitter. They adopt the COSMOS Framework for collaborative system based on MPEG-4 Objects and Streams and Java 3D for graphics rendering as well as for creating and manipulating 3D objects.

Sung et al. [114] developed a networked haptic game based on their own data structures and performed some experiments to analyze the characteristics of haptic data transmission in real system as well as to examine which existing protocol (TCP or UDP) is better for haptic interactions. The experiments conclude that a UDP structure is more efficient than a TCP for haptic data transmission.

Hubbold [115] described a prototype implementation of a complex task in a shared, collaborative virtual environment – a simulation of two people carrying a stretcher. Inter-program communication was implemented based on TCP and UDP on the UNC Virtual Reality Peripherals Network (VRPN) libraries.

Caroline et al. [116] analyzed how users are affected by delayed haptic force feedback from other people. They also proposed an "impact-perceive-adapt" model of user performance to explain the observed pattern of performance, which considers the interaction between performance measures, perception of latency, and the breakdown of perception of immediate causality.

Vaghi et al. [117], presented qualitative observations in an experimental shared environment with increasing network delays and get the conclusion that more awareness of the behaviour to the system can improve the participants' performance.

3.3 Summary

In this chapter, classifications and comparisons of networked visualization and collaboration are discussed. Model transmission is concluded to be more suitable for networked visualization and collaboration. Approaches for networked haptic interaction and collaboration are also classified and compared; specific problems for networked haptic interaction and collaboration are then further demonstrated. Although networked collaborations have been developing fast with the explosion of Internet, we are still facing critical challenges in both visual and haptic collaboration. The network latency seriously restricts the development of collaborations and such restriction could not be totally solved in the near future due to hardware limitations. Therefore, reducing the size of the models transmitted over the network would be the most feasible approach to support interactive networked collaborations, especially when both visual and haptic collaboration are involved. Based on the survey from this chapter, a conclusion could be drawn that an efficient solution for networked collaboration is in urgent need to provide reduced data size for efficient transmission while still keep good rendering quality and fast interaction.

Chapter 4 Uniform paradigm of visual and haptic modeling and rendering

In this chapter, a uniform approach to visual and haptic modeling and rendering is proposed. In Section 4.1 we propose the uniform modeling paradigm which supports integral and consistent definition of various geometry, visual appearance and tangible physical properties. In Section 4.2 we introduce function-based approach to further enhance the compactness and flexibility of the approach. The algorithms for haptic interaction are presented in Section 4.3. Finally in Section 4.4 the summary is given and further research on haptic collaboration is detailed.

4.1 Unified modeling paradigm

As previously reviewed in Chapter 2 about visual and haptic rendering, the two rendering pipelines are heavily competing for CPU resources. We aim to solve such problem by proposing a unified modeling paradigm.

In the real world, different properties of a single object are appreciated through different human senses. This procedure was simulated on computer by property separation. For example, separation of geometry and visual appearance is a common approach in different web visualization and computer graphics data formats and software libraries. Following this routine, we propose a concept of independent definition of geometry, appearance and haptic properties.

In the meantime, since our senses are complementary, the senses from separated properties are finally merged together to be conceived as one object. Therefore in our

unified modeling paradigm, geometry, visual appearance and tangible physical properties are also first separated and defined in their own coordinate domains and then merged together to constitute a virtual object.

In the visual rendering pipeline, at least one virtual camera is associated with the observer and each visible 3D point is displayed on a 2D image with a certain colour. Different illumination models can be used to obtain colours, which are commonly pseudo-physically based. For example, Phong Illumination model is defined by ambient, diffuse and specular reflections without physical basis. It can be augmented by transparency property of the object. Although Phong Illumination model has several assumptions and properties in the model are artificially classified, it can represent all illumination situations and therefore commonly used in computer graphics. Geometry samples the visual properties and therefore they have to be associated with geometry of the object however it may be rather a placeholder or bounding surfaces, visible or invisible, for objects like clouds and fire. The geometry can be either a curve, or a surface or a solid object.

In the haptic rendering pipeline, we need physical properties of the object which will produce forces at each point of the virtual environment that we can perceive through a haptic force-feedback device. The number of force vectors which can be calculated for any given point in 3D space depends on the haptic device used. Thus it can be one force vector for translational force or two vectors for translational and torque forces. These forces can be exerted when the user examines the outer or inner part of the object, or when the moving object collides with the haptic tool, or when a force field is produced by the object, as well as by any combination of the above. Like visual properties, the

physical properties are also associated with some geometry which we call “haptic container”, however it can be rather a placeholder or even an invisible container for some physical properties. To explore surface friction and density, the virtual representation of the actuator, which can be defined as a virtual point, vector, frame of reference or a 3D object, has to be moved by the surface or inside the object. To explore the force field, it is sufficient to hold the actuator within the area where the force is exerted.

Modeling of the three components of the objects—geometry, visual appearance and physical properties—can be performed either in the same coordinate system or in different coordinate domains which will then be mapped to one modeling coordinate system of the virtual object. The domains are defined by the bounding boxes (Figure. 4.1) which specify their location and coordinate scaling. Hence, each of the components can be defined in some normalized domain and then used for making an instance of an object with different size and location. For each of the components, an individual instance transformation has to be defined. The resulting object is obtained by merging together the instances of geometry, appearance and physical properties. This approach allows us to make libraries of predefined geometries, appearances and physical properties to be used in different applications.

The paradigm we proposed is not overcomplicated but it covers most scenarios for both visual and haptic rendering. For example, plenty of physics-related research and applications use mass and gravity as key components for rendering and animations. We do not propose such components in our modeling paradigm because the haptic rendering result for mass and gravity could simply be considered as a special case of force field rendering.

Graphics rendering gives us more information as it shows the scene all at once represented by numerous pixels constituting the image, while haptic rendering is mostly a dynamic process where we acquire and collect information about objects in the form of a force feedback by moving haptic actuators on and inside the objects. Therefore an object that can be used as a 3D visual representation (avatar) of the virtual haptic tool is needed to follow the user's movements, identify the current transformation information (translation, rotation, scale) in the virtual environment, as well as to haptically interact with virtual objects. The way of displaying the haptic tool should be easily changed by the user according to the scene content. The way how the haptic tool is associated with the haptic device should be flexible since most haptic devices have limited angles of rotations for their actuators and may need to reattach the haptic tool to reach the point of interest at a required angle.

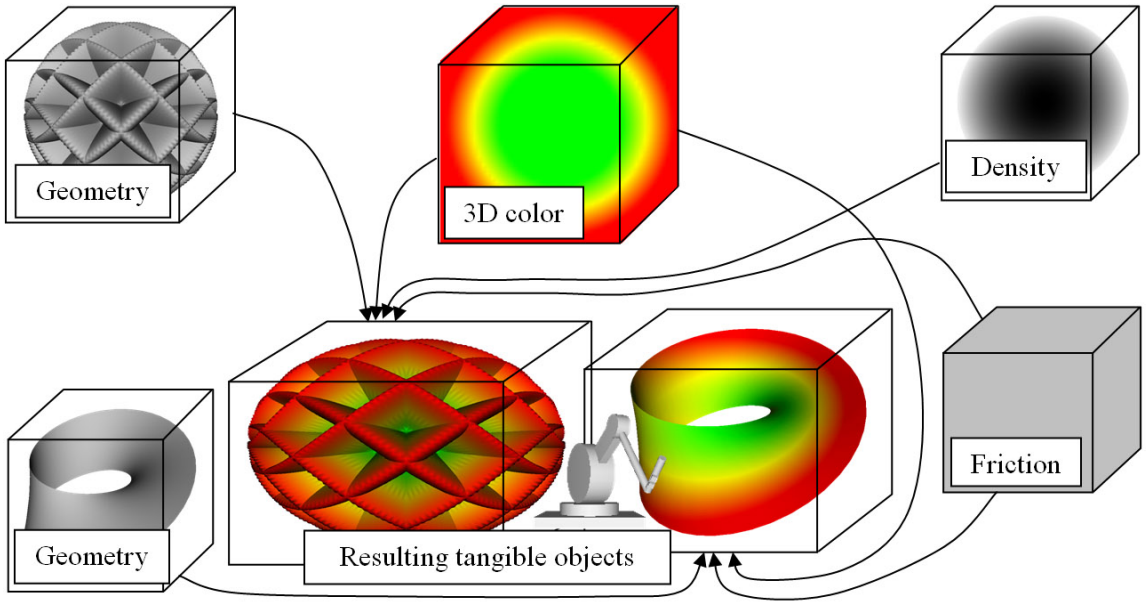


Figure 4.1 Assembling geometric objects from geometry, appearance and physical properties defined in their own coordinate domains

4.2 Function-based approach

Our unified modeling paradigm is able to handle various objects properties, however the size of the property definitions could be very large, which makes them very inconvenient to store and transfer. Therefore, we propose to improve the modeling paradigm by using function-based approach.

The “Function based approach” means using mathematical functions and procedures for definition of geometry, appearance and physical properties of the objects. The functions in our approach serve both as descriptive language as well as implementation algorithms. This approach allows us to be flexible with choosing the actual visualization platform. Mathematically there are three ways of analytical function definitions: explicit, implicit, and parametric functions. It is quite common that only one of these ways is considered when functions are used in shape modelling and visualization. On the contrary, in our approach we allow to mix these three definitions to achieve the most efficient representation of the objects.

The origins of our approach lie in the work [97] where so-called F-Rep representation was proposed in 1995, however the problem in this paper was restricted to geometry domain. Later in 2006, Liu et al. [118] expanded the problem scope to visual appearance domain by using explicit and parametric functions to represent 3D colours and 3D geometric textures. In this PhD thesis, the author has further expanded the problem scope to physical property domain and proposed to use functions to both represent and implement various physical properties such as tension, friction, density and force field.

Some other haptic rendering research papers have also proposed to use mathematical

formulae in their approach, such as bone surgery simulation [119, 120, 121], electrocautery procedures simulation [122], and breast palpation [123]. While their research topics are aimed at simulating specific operations and scenarios with particular haptic tools, this PhD project adopts the function based approach to provide a uniform modeling paradigm to represent and implement various object properties such as geometry, appearance and physics in a non-specific way. Compared to other haptic rendering research papers, whose main contributions are complex and problem-specific formulae which provide haptic rendering to fixed configurations, the function based approach proposed in this PhD project could expand plenty of existing virtual environments with uniform representations of geometries, appearances and physical properties, and render them within one paradigm. The use of the "function based approach" is not to overcomplicate the modelling framework but to provide an intuitive and fast approach to build haptic-enabled scenes with various and accurate physical properties. Besides, since the function based approach applies to various aspects of object properties in a uniform way, other object properties such as hearing and smell could also be easily incorporated into the existing function based modeling paradigm to provide better immersion. In contrast, problem-specific approaches, which also incorporate mathematical formulae, may require major architecture revision to properly define and render the extra components.

In computer graphics, it is generally recognized that a virtual object could be considered as a combination of geometry and visual appearance. Since both visual and haptic rendering are considered in this thesis, the thesis author follows the routine and proposes to separate object properties into three components: geometry, appearance and

physics. These components could be further classified into more detailed components: geometry could be defined as curve, surface and solid, appearance could be defined as 3D colour and 3D geometric texture, and physics could be defined as surface property, density and force field.

Figure 4.2 shows which combinations between object property (geometry, appearance and physics) and function type (implicit, explicit and parametric) have a practical application.

Geometric curves can be conveniently defined by parametric functions as: $x = f_1(u|,t)$, $y = f_2(u|,t)$, $z = f_3(u|,t)$. Geometric surfaces can be defined by either using implicit function *f(x,y,z,t) = 0* or by parametric functions as: $x = f_1(u|,v|,t)$, $y = f_2(u|,v|,t)$, $z = f_3(u|,v|,t)$. Solid geometry could be defined using explicit function *g = f(x,y,z,t) ≥0*, or defined by parametric functions $x = f_1(u|,v|,w|,t)$, $y = f_2(u|,v|,w|,t)$, $z = f_3(u|,v|,w|,t)$. Addition of time t allows for definition of animated objects.

3D colour can be defined as explicit functions *g = f(x,y,z,t)*, followed by colour interpolation to get the respective RGB values or directly defined by parametric functions $r = \varphi_1(u|,v|,w|,t)$, $g = \varphi_2(u|,v|,w|,t)$, $b = \varphi_3(u|,v|,w|,t)$. 3D geometric texture can be defined by distance functions either explicitly as *displacement = f(x,y,z,t)* or parametrically $x = f_1(u|,v|,w|,t)$, $y = f_2(u|,v|,w|,t)$, $z = f_3(u|,v|,w|,t)$.

Surface physical properties (including tension and friction), as well as density, can be defined as explicit functions *g = f(x,y,z,t)*. For each point on the object surface and inside the object, a corresponding tension, friction and density value will be calculated and rendered. Force field property can be defined using parametric functions

$x = f_1(u|,v|,w|,t)$, $y = f_2(u|,v|,w|,t)$, $z = f_3(u|,v|,w|,t)$ at each point inside the force field, a corresponding force vector *(x,y,z)* will be generated to push the haptic handle to a new point.

In Figure 4.2 the part within dashed line is the classifications of geometry, appearance and physics, while the part within the dash-dot-dash line is possible meaningful combinations among various object properties and various mathematical functions.

Combinations of the representations that were not discussed above are not excluded from our consideration. They just have not been utilized for meaningful representations. In the future, we may further improve the paradigm to introduce new components as well as propose more meaningful combinations among them.

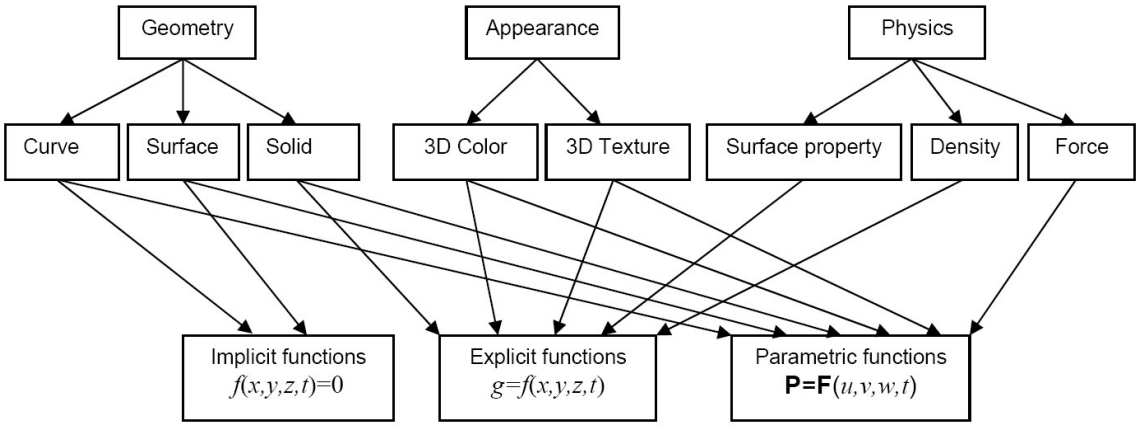


Figure 4.2 Definition of geometry, appearance and physics of shapes by implicit, explicit and parametric functions

4.3 Uniform algorithm of haptic interaction

We aim to define surface and solid physical properties as well as ubiquities forces in virtual scenes with mixed geometric models, including polygon meshes, point clouds,

image billboards and layered textures, voxel models and functions-based models of surfaces and solids. We are also going to propose a way how to identify location of the haptic tool in such haptic scenes as well as consistently and seamlessly determine physical properties when the haptic tool moves in the scenes with objects having different sizes, locations, and mutual penetrations.

To introduce the physical properties to the mixed geometric models, we propose that each physical property must have a certain geometric container within which this property can be haptically rendered. Such a container may be a geometric surface of the object which is augmented with the physical property (Figure. 4.3a). It can also be an invisible surface specially defined for the physical property and not necessarily coinciding with the geometric surface of the object (Figure. 4.3b and 4.3c).

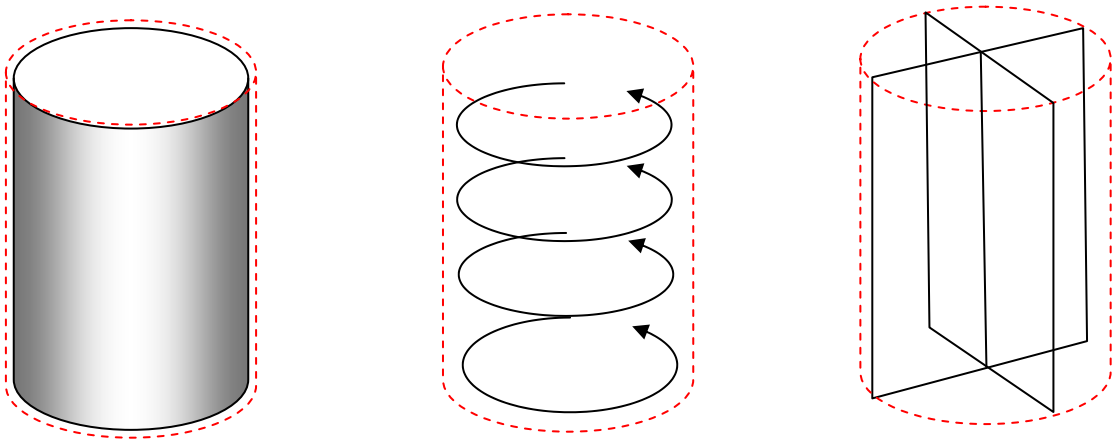


Figure 4.3 Haptic containers (dashed line): (a) Actual surface of the object (b) Surface containing haptic forces (c) Surface defining an object rendered without showing the actual surface (billboards, layered textures, point clouds)

This approach allows us to create physical properties for objects which do not have any surface at all such as point clouds or objects displayed as layered textures (e.g. MRI images). This approach also allows us to add ubiquities forces to the scenes by

encapsulating them into the invisible haptic containers. For each of the haptic container, be it a real surface of the object or a specially defined surface for the physical property, we allow for concurrent definition of surface properties (tension and friction), solid properties (density), and force fields.

To be able to consistently and seamlessly switch between the surface and inner solid and force properties, we revise the concept of surface stiffness commonly used in haptic SDKs, which defines how fast the force vector will increase as the haptic tool penetrates beneath the object surface. The result of such surface force generation is unrealistic since the user always feels the surface (Figure 4.4a) rather than the inner viscosity when both properties are defined. On the other hand, when viscosity is defined, no surface rendering can be performed (Figure 4.4b). To provide for haptic rendering of both surface and inner physical properties, we propose to introduce a certain surface zone within which the surface tension and friction properties will be activated while still allowing the haptic tool to penetrate the surface and render the forces resulting from the inner physical properties (Figure 4.4c). The depth of the surface zone may depend on the precision, workspace and maximum force of haptic device. Besides, the depth of the surface zone could also be defined by mathematical functions to better simulate complex surface properties of real life object (Figure 4.4d).

In contrast to the commonly used predefined location of the haptic tool and prior knowledge of the physical properties to be rendered, we propose an algorithm for identifying the location of the endpoint of the haptic tool—Haptic Interface Point (HIP)—in the virtual environment. Our approach is based on casting multiple haptic rays in different directions from the current position of the HIP as soon as the tool arrives in

the scene (Figure 4.5). The rays are cast to find the haptic container within which the HIP is located. The respective physical property will then be rendered. The number and the length of the rays can be adaptively adjusted by the local complexity of the scene, as well as manually by the user.

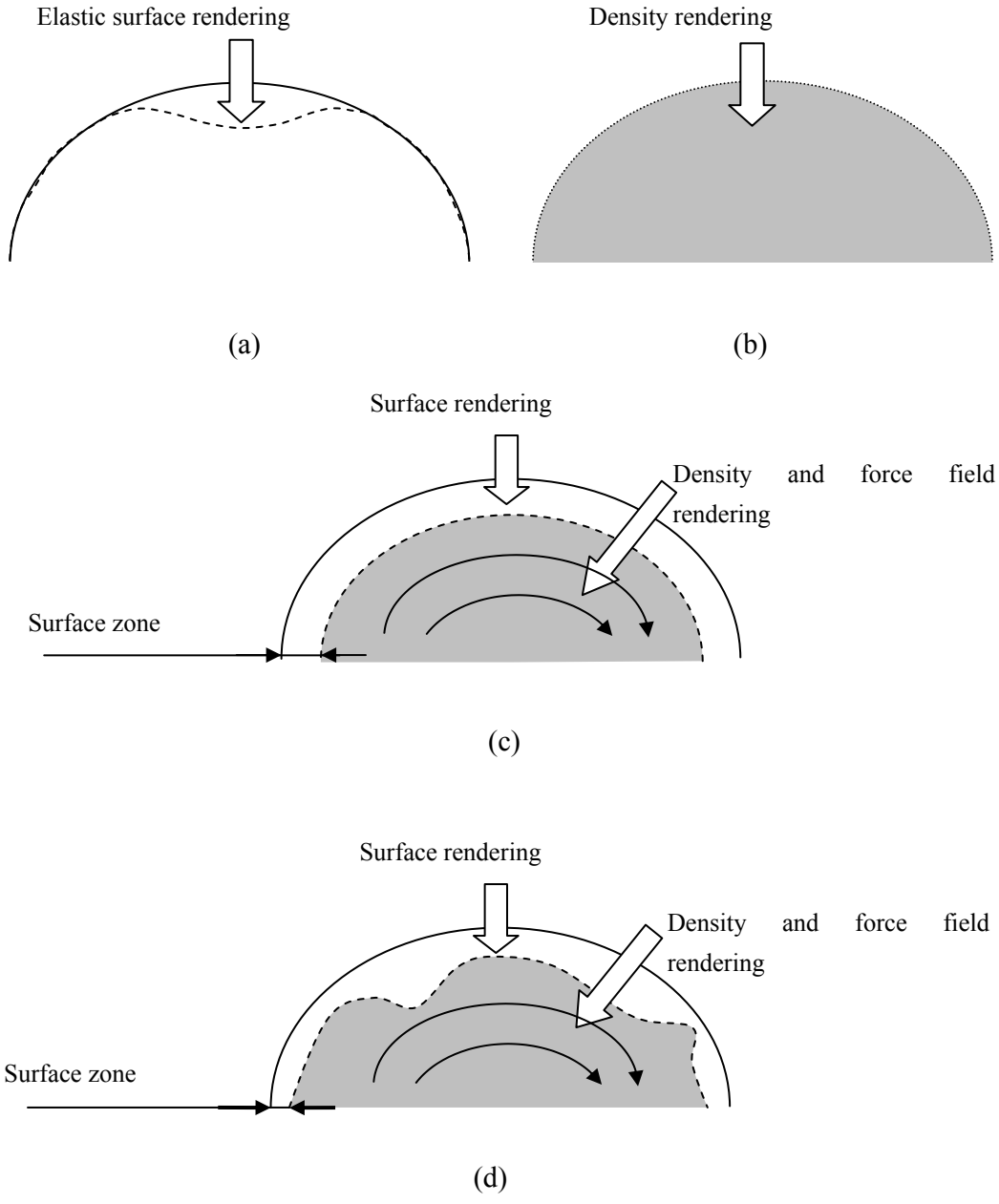


Figure 4.4 Common surface (a) and viscosity (b) rendering (c) Combined surface, viscosity and force field rendering with the surface zone defined, and (d) surface zone defined by mathematical functions

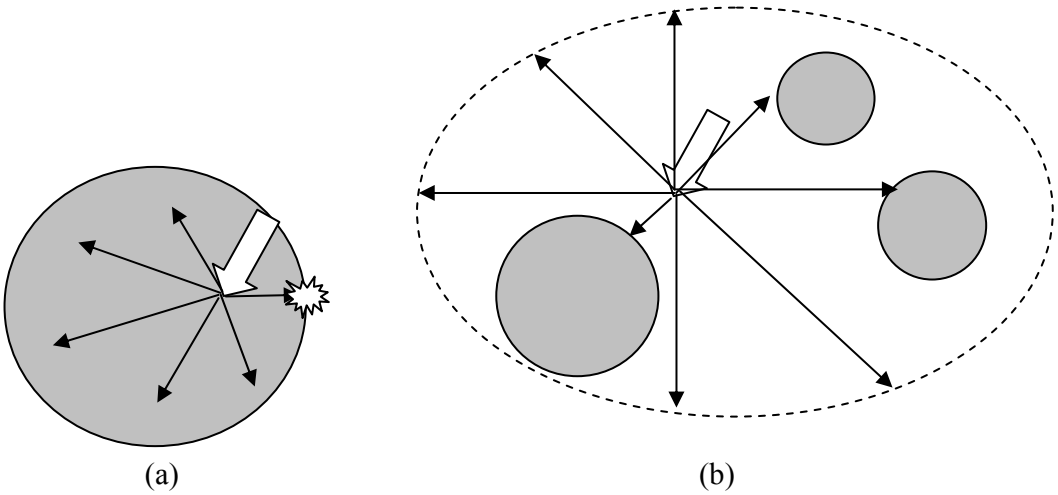


Figure 4.5 Casting haptic rays to detect the virtual tool location: (a) the tool is inside a haptic object, (b) the tool is close to several objects and it is also inside another larger object. The inner haptic property of the larger object will be rendered

When the tool moves through the scene, in contrast to the common approach of colliding with polygons using one ray following the direction of haptic stylus [124], we suggest a continuous multiple-ray casting to switch between the three types of physical properties (surface, solid, and force field). Compared with [124] which requires a pre-computation phase to build hierarchical database of the scene, our approach does not require such prior knowledge of the scene, while it could still immediately identify HIP location and start force rendering on the fly. We also propose a stack-based algorithm to permit mutual penetrations of the objects with different physical properties defined (Figure 4.6).

The proposed stack-based algorithm is aimed to solve haptic rendering problems with mutually penetrating objects. In real life it corresponds to placement of one solid object inside another solid object, i.e. in medical applications it can be an implant or a lump surrounded by human body tissues.

In haptic rendering, most commonly used approaches are based on polygon-based

models, which only have object surfaces defined. Therefore the situation of mutually penetrated objects does not need to be considered. On the other hand, haptic rendering with solid models has also been deeply studied. Since solid models have interior properties defined, a question has been raised of how to correctly render haptic effects when two or more solid objects mutually penetrate, as their overlapping parts may contain different haptic effect definitions from several objects.

Hence mutually penetrating objects require a proper solution and we propose three options for it:

1. The haptic effect for the overlapping part is determined by the last object the haptic handle has penetrated.
2. The haptic effect for the overlapping part is determined by intersection operation of some or all of the involved objects.
3. The haptic effect for the overlapping part is determined by union operation of some or all of the involved objects.

Depending on the content creator's intention, any of the three options could be adopted. However the author chooses to adopt option one and implemented it as a stack-based algorithm. The author does not exclude though other options and any of them can be easily incorporated into the existing haptic collision and rendering framework.

The algorithm builds a stack to keep track on all colliding objects with the Haptic Interaction Point (HIP). When the HIP penetrates a new solid object, the object name will be pushed into the stack; when the HIP leaves the solid object, the object name will be removed from the stack. At any moment, the current rendering haptic effect will be retrieved from the top most object in the stack. This makes sense in most cases however

there can be some confusing exceptions. For example, if the HIP penetrates a series of objects twice from different directions, the last object the HIP penetrates may be different; therefore the rendering result for the most internal object may also vary. Such a problem also exists in visual rendering when two solid objects with different colours mutually penetrate. The "correct" colour for the overlapping part is questionable and usually is set by the programming system.

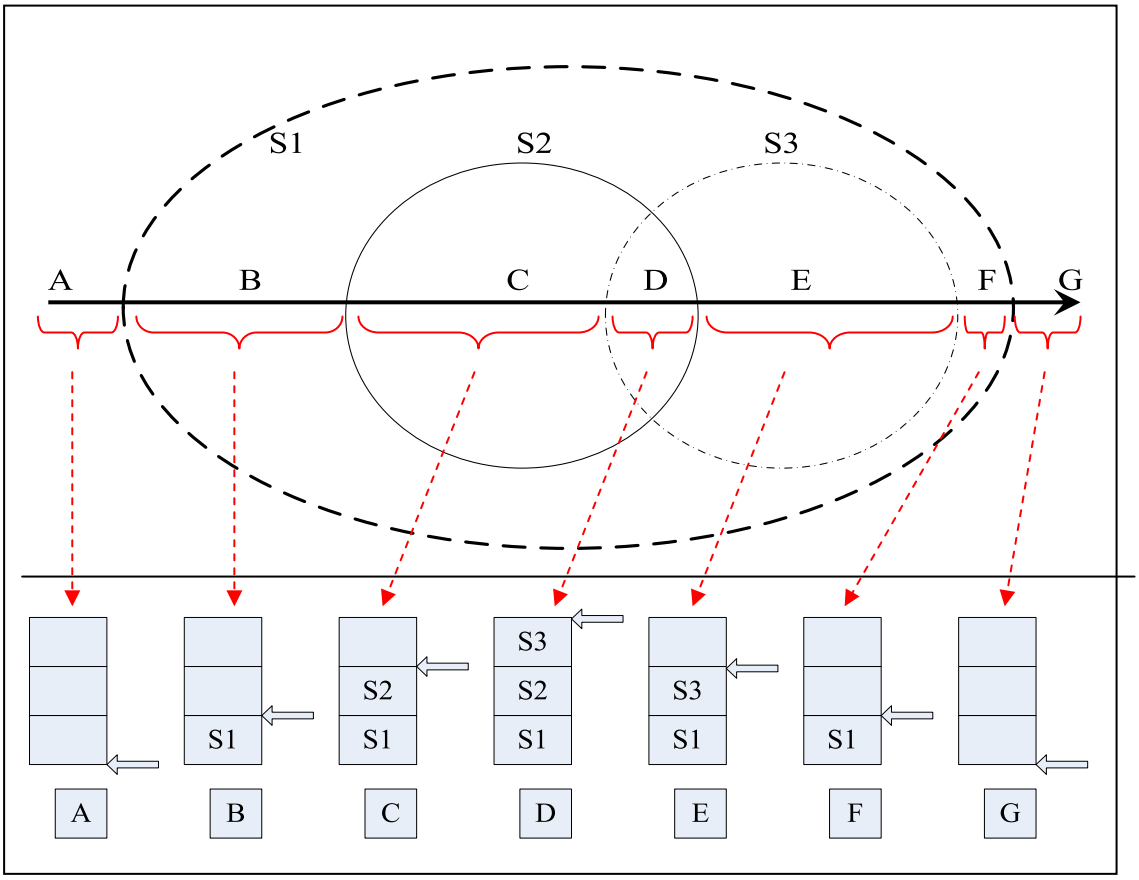


Figure 4.6 Inclusion & Intersection graph and stack analysis

In contrast to the commonly used predefined location of the haptic tool and prior knowledge of the scene and haptic effects to be rendered, the stack-based algorithm does not require such prior knowledge and could still identify HIP location and haptic effects to be rendered on the fly. In Figure 4.6, there are three objects S1 (dashed line), S2 (solid line) and S3 (dash-dot-dash line) defined with different haptic effects on their surfaces as

well as inside them. S1 is the biggest object which includes mutually penetrating S2 and S3. The HIP moves along the direction indicated by the bold line from left to right and collides with the surfaces of the three objects six times. A stack is constructed to record collision history with all objects. A pointer which always points to the stack top is also used to indicate current haptic effects. While the HIP is at segment A, it has not collided with any of the objects, therefore the stack is empty and no haptic effects will be rendered. While the HIP is within segment B, it has penetrated S1 therefore S1 was put into stack. HIP is now inside S1 but still outside S2 and S3, and haptic effects of S1 should be rendered. While HIP is within segment C, it has penetrated S2 and therefore S2 was put into stack. Current HIP is within both S1 and S2 but S2 is the last penetrated object, therefore haptic effects of S2 interior will be rendered. While HIP is within segment D, it has penetrated all three objects and S3 is the last penetrated object, therefore haptic effects of S3 should be rendered. While HIP is within segment E, the HIP has already left S2 and therefore S2 was removed from the stack. While HIP is within segment F, the HIP has left S3, and S3 was removed from the stack. The pointer is now pointing to S1 with the haptic effects we need to render (the HIP is still within S1). While HIP is within segment G, the HIP has left S1 and the stack was cleared, therefore no haptic effects will be rendered.

Usually, most haptic scenes are still not as complicated as visual scenes. Typically, haptic scenes only have several objects with haptic effects at predefined positions. Therefore mutually penetrating objects were generally avoided or not specifically processed in most haptic collision and rendering algorithms and SDKs, which will be discussed below.

In [44], Ruspini proposed the idea of using a hierarchical bounding sphere for haptic collision detection. GAMMA (Geometric Algorithms for Modeling, Motion, and Animation) Research Group from University of North Carolina designed and implemented H-Collide [45], which adopts spatial subdivision and OBB trees for haptic collision detection. In [54], Zilles and Salisbury first introduced a name of the haptic proxy as “god object” and then extended the “god object” idea and proposed approaches to implement force shading, surface stiffness, friction, and haptic texture. However, all these haptic collision and rendering algorithms are based on polygonal meshes and they do not consider situations with mutually penetrated solid objects.

There are also several haptic collision and rendering algorithms available for solid objects. In [62], a voxel sampling approach has been proposed to accurately render complex real-world task. In [63], a solution was proposed to haptically render the volume data using dynamic spline-based implicit functions. In [64] Kim et al. proposed a method of touching a virtual volumetric non-rigid object with the object at multiple points. In [65, 66] Lundin et al. proposed proxy-based haptic interaction with volume MRI data based on domain knowledge filtering to extract separate material and surface information and to avoid explicit classification. The result has been used on SenseGraphics (Volume Haptic Toolkit) VHTK [67] which is a volume haptic toolkit comes with functionality for scalar and vector visualization methods and includes haptic modes for surface, gradient and viscosity force rendering. However, these algorithms also did not discuss the mutually penetrated situations and provide solutions.

Besides, there are several haptic SDKs available: Reachin API&HaptX, CHAI3D, OpenHaptics toolkit, Novint SDK and H3D&HAPI.

Reachin API utilizes C++, Python and VRML to provide users with a flexibility of implementing haptic programs. HaptX is a haptics engine from Reachin, which works as a haptic toolbox for easy and fast development. Although in [125] the authors claimed to have extended Reachin API and implemented for volume rendering, there is no example available in their free API demo to show the ability of rendering solid objects. Since we have not bought the Reachin API license, we are yet to know how it will work with the mutually penetrated situations. However based on discussions with other researchers we estimate it will work similarly to other commercial haptic SDKs.

CHAI3D is an open source SDK that supports haptic devices from different vendors. It adopts OpenGL for visualization and ODE (Open Dynamics Engine) for rigid body collision detection.

Open Haptic Toolkit from Sensable provides both High-Level and Low-Level haptic collision detection and rendering APIs to different users. Open Haptic Toolkit support rendering of both polygonal and solid objects.

Novint SDK is similar to HDAPI in Open Haptic Toolkit however it is simpler and provides less API functions.

H3D API is a scene graph based API by SenseGraphics Company, which is based on X3D, C++ and Python to render the scene both graphically and haptically. HAPI is also developed by SenseGraphics and is intended to be a haptic rendering engine. It is device independent and purely built with C++. It supports OpenHaptics toolkit/CHAI3D/God Object/Ruspini for haptic collision detection and force rendering.

All the above-mentioned haptic SDKs do not provide procedures to handle scenes with mutual penetrations, and therefore, for example, when using them to build a scene

with one small sphere located completely inside another big sphere, effects from both spheres will be rendered when the handle is inside the small sphere. This may be reasonably explained but could be unrealistic for certain applications, where only properties from the small sphere need to be rendered. Theoretically, users could implement their own algorithms based on these SDKs to solve the problem of mutual penetration, such as the stack-based algorithm proposed and implemented in this PhD thesis.

There are also several other haptic collision and rendering algorithms which deal with mutually penetrating objects but they are based on volume penetrations and only aimed at generating forces to avoid actual mutual penetrations. In [57, 126], a new data structure called inner sphere tree was proposed to support proximity query and penetration calculation at 1000 HZ. These algorithms are not aimed at solving the mutually penetrated problem anyway.

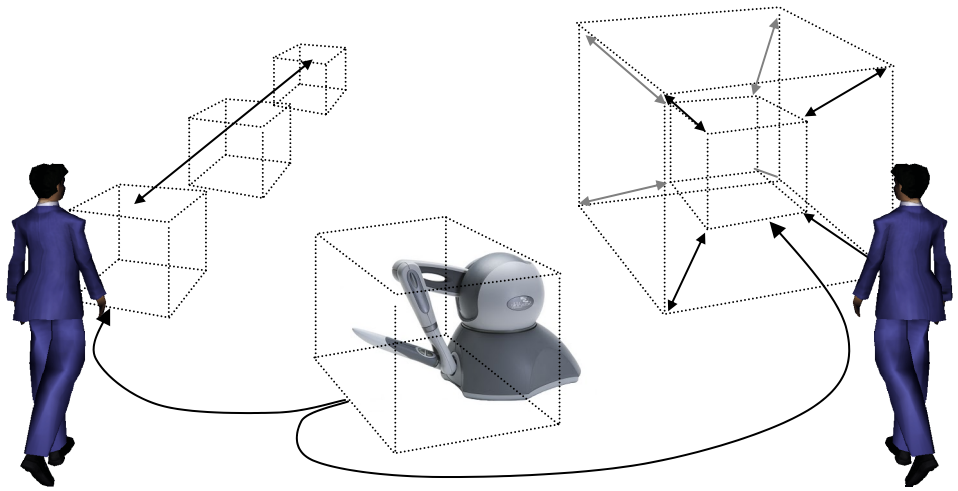


Figure 4.7 Movable and resizable 3D haptic zone

To be able to reach different parts of the scene with a different haptic precision, we introduce a movable and resizable 3D haptic zone which maps to the actual workplace of the haptic device. This approach corresponds to an expandable virtual arm which size can change as well (Figure 4.7).

4.4 Summary

In this chapter, a uniform approach to visual and haptic modeling and rendering is proposed, which could handle various geometry, visual appearance and tangible physical properties. The integration of function-based approach further enhances the performance and provides more flexibility to the modeling procedure. The algorithm for haptic interaction is presented to show advantages of the proposed approach. The uniform approach could be used not only on a single computer but also in collaborative environments. Therefore it will be further extended in the next chapter to support the proposal of visual and haptic collaboration.

Chapter 5 Framework of visual and haptic interaction and collaboration

In this chapter, a novel visual and haptic collaborative framework is proposed. In section 5.1 we discuss general issues for 3D networked collaboration. In Section 5.2, highlights of the proposed visual and haptic collaborative framework are discussed in detail, including the function-based approach which provides more flexible object definition and faster data transmission. In Section 5.3, the general architecture and workflow of the proposed framework are demonstrated. Finally in Section 5.4, we make a conclusion about the research and implementation values based of the proposed framework.

5.1 General issues for 3D networked collaboration

There are several functions typically required to be implemented in 3D networked collaborative applications, namely *information transmission*, *shared objects*, *shared events*, *synchronization*, and *consistency control*.

To support both visual and haptic collaboration, *shared objects* with physical properties defined have to be used.

Ideally, when a virtual scene is set to be shared among several users, each object has to become a shared object, i.e. changes of its location, geometry and appearance have to be synchronously seen by all the users in the scene. However, in reality only a limited number of objects in the scene are declared shared because it requires frequent transmissions across the network of events about the location and orientation of such objects as well as their model definitions when they change. In many applications, mostly

location and orientation of shared objects are considered (e.g. shared virtual scenes with the objects available in virtual stores).

The scene is normally shared by downloading it to the client computer, and by running locally different scripts controlling object behaviour that can be synchronized by either timer signals common for internet connected computers or small events sent between the clients and/or server. In fact, in each shared virtual scene there are always a few shared objects which are visual avatars of the users however they are stored on the client computer or web-located objects, which URLs are provided to the viewer via different ways (e.g. scripts, html pages, and databases associated with the scene).

We define a *shared object* as the object which visual and physical properties (not only location and orientation) are synchronously rendered by all the client computers as they change.

Same as for a single computer haptic interaction, there is also a need to define a 3D visual representation (avatar) of the virtual haptic tool. In our considerations, such object is just a shared object that is associated with the respective haptic device to change its location and orientation and possibly even geometry and appearance while it is being applied. The motion of this object can be seen by all the participants synchronously and it can interact with other objects that possess tangible physical properties.

Shared objects are normally used together with *shared events*. Shared events are inherited from event transmission where an event from one client is sent to other clients either through server or directly. No matter which way the platform adopts, *synchronization* is performed to allow multiple users to immerse into the virtual scene and share it. Another crucial issue here is how to prevent concurrent operations on shared

objects to be performed by different clients. To ensure *synchronization* amongst all clients, *consistency control* algorithms such as locking and serialization mechanisms are required.

Shared objects can be *locked* by the user for their private use. Hence, a shared object selected by the user as a tool can be then set as unavailable for other users. They will be able to see how this object moves following the motion of the user's haptic device actuator. However, we permit use of a shared object as a tool by several users. The motion of the object will then be a resulting motion of the haptic devices connected to it. *Locking* can also be useful when interactive changes to the object are being performed by one of the users while others are not expected to make any changes to it. Last but not least, each user may require more than one haptic device with different tools associated and used concurrently. These can be either different haptic devices or devices with multiple actuators, which can be considered as different virtual tools in the scene. Strong locking means that the client has to release the lock before another client can succeed in requesting the lock. Weak locking indicates that if the client requests the lock while another owns it, the owning client loses the lock and the requester gains the lock.

5.2 Highlights of the proposed collaboration framework

We introduce two modes for communication using text strings in the proposed haptic collaborative framework. The first mode considers user input as plain messages and directly relay them to all clients. The second mode considers the user input as commands that assist with visual and haptic collaboration. By recognizing predefined keywords in the text, the user input could interactively redefine properties of shared objects, load new shared objects from external libraries, etc. This communication mode is crucial to the

proposed framework. When the user changes the property of a shared object, a message containing the updated definition will be generated from the local client and propagated among the collaborative environment to ensure all other clients have received the message. However, according to the contents in the message, only qualified clients will be updated.

We introduce two approaches to store environment state in the proposed framework. The first approach relies on the server to always keep a copy of the up-to-date environment state. By using such an approach, the environment state will always be available even when no clients are connected to the server. The second approach depends on the first client logged in to the server. All other clients will be synchronized to the environment state that the first client kept when they log in. If the first client has left, the server will automatically assign another client to do such synchronizations from all new coming clients. The server state will be lost when all clients have left.

We introduce comprehensive routines to convert all modifications of visual and haptic properties into string messages as well as force interactions among different clients. For visual and haptic property modifications, definition of the updated property will be composed and sent out for rendering, while for direct force reaction among clients, the counter forces on each other client will be calculated and sent back based on Newton's Third law of Motion.

Visual and haptic property modifications can be carried out by the combined use of haptic operations and user input commands. First, the user needs to select a property to modify. This is done simply by typing a command in the command window. After that, the user will be able to set a desired property definition for his tool avatar, which will

replace the default property definition of the shared object after modification. The user can then start the collaborative modeling procedure on the shared object by overlapping his tool avatar with the shared object. By confirming the property modification, corresponding haptic property definitions will be applied onto the overlapped part of the shared object. In the mean time, all other clients will be synchronously updated with such property modifications, whether it is visual or haptic. This procedure provides the freedom of modeling shared objects with sophisticated visual and physical properties, which could not be done by traditional approaches using very complex scripts.

Since the transmitted messages are usually large in size, we introduce special routines for splitting and recomposing long messages so that they could be split into blocks before transmission and recomposed on the destination client. Ideally, depending on the network bandwidth, the size of the message block should be configurable. Long message block works better with good network bandwidth, but if one block is not transmitted correctly, the resending procedure will take longer time. Short message block works better with poor network bandwidth, but it will take more time to split and recompose them. Each client maintains a pair of Sender and Receiver daemons, which always monitor the input and output of local message pool. The message strings are dissembled and assembled across the collaboration platform as well as among different local nodes to ensure the information will be transmitted in the right order and executed in the supposed way. The message strings are self-explanatory; they have all necessary information tags such as sender and receiver names, collaborative session ID, type code, operation code, counter, position, angle, etc. However, not every message string will need all of this information. Only necessary data will be included to avoid jitter and delay in

network transmission. The last message block contains the end tag and therefore the receiving client will be notified that all blocks of the message have been received. Since errors may happen during transmission, if one message block is invalid, a requirement from the receiving client will be delivered to the sending client asking for resending the specific message block.

The performance of collaboration largely depends on the size of the models transmitted over the network, especially when the 1000 HZ haptic interaction is required. Therefore, a faster message transmission mechanism which largely reduces the transmitted data size is incorporated into the framework. Such a mechanism is also based on mathematical functions and procedures, which is naturally compatible with the unified modeling paradigm. Therefore each aspect of the object properties will be easily converted into a message. Small function-based models can reduce the error rate during transmission, provide any required level of details and significantly augment, enrich and replace traditional large polygon and point-based models which are usually used for collaboration.

An example is shown below to illustrate the transmission mechanism using mathematical functions. In the proposed collaborative environment, there is an existing shared object defined as follows:

Geometry: *1-x^2-y^2-z^2*

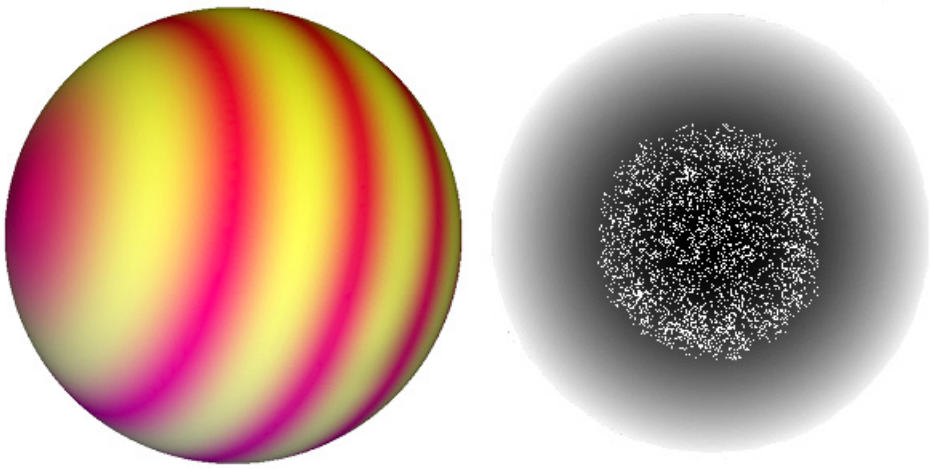
Colour: *r=t; g=abs(sin(5*u*pi)); b=w;*

Density *(script definition): function frep(x,y,z,t){dist=sqrt(x^2+y^2+z^2);*

*g=0.8; a=120*pi; if(dist<0.7) g=0.3+0.2*sin(x*a)*sin(y*a)*sin(z*a)); return g;*

The geometry is defined as a sphere. The colour is defined as time-dependent

(animated) parametric functions (Figure 5.1a). The density is defined using script as a distance function and a noise function, which changes from a hard shell to a crunchy core (Figure 5.1b).



(a) (b)

Figure 5.1 (a) Time-dependent colour defined by parametric functions (b) Non-uniform density defined by function script

Two users are concurrently working with the shared object in a collaborative session. User A intends to change the geometry and adds 3D geometric textures to the object, while User B plans to add tension and force field to the object. They will model the shared object using haptic devices for interaction and mathematical functions and procedures as object descriptions, which will generate corresponding messages to be transmitted within the collaborative environment:

User A: changes object geometry to:

*x=cos(u*pi)*((sin(0.5*v*pi))^3);y=sin(u*pi)*((sin(0.5*v*pi))^3);z=cos(0.5*v*pi);*

The corresponding message generated:

"updated geometry definition:

BEGIN

*x=cos(u*pi)*((sin(0.5*v*pi))^3);y=sin(u*pi)*((sin(0.5*v*pi))^3);z=cos(0.5*v*pi);END"*

User A: adds 3D geometric texture:

0.2(sin(10*pi*x)*sin(10*pi*y)+sin(10*pi*x)*sin(20*pi*z)+ sin(10*pi*y)*sin(20*pi*z))*

The corresponding message generated:

"updated texture definition:

BEGIN 0.2(sin(10*pi*x)*sin(10*pi*y)+sin(10*pi*x)*sin(20*pi*z)+ sin(10*pi*y)*sin(20*pi*z)); END"*

User B: adds object tension to:

function frep(x,y,z,t){ if (y>-0.9) g=x^2+y^2+z^2; else g=0.4; return g;

The corresponding message generated:

"updated tension definition:

BEGIN function frep(x,y,z,t){ if (y>-0.9) g=x^2+y^2+z^2; else g=0.4; return g; END"

User B: adds object force field to:

*function parametric_x(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return x/sqrt(x^2+y^2+z^2); }*

*function parametric_y(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return y/sqrt(x^2+y^2+z^2); }*

*function parametric_z(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return z/sqrt(x^2+y^2+z^2); }*

The corresponding message generated:

"updated force field definition:

*BEGIN function parametric_x(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return x/sqrt(x^2+y^2+z^2); } function parametric_y(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return y/sqrt(x^2+y^2+z^2); } function parametric_z(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return z/sqrt(x^2+y^2+z^2); } END"*

Such messages will be distributed to all clients in the collaborative environment and re-interpreted as commands. Therefore, all clients will be updated to the latest collaboration status. After the collaborative modelling session, the shared object has a new geometry, as well as 3D geometric texture, surface tension, and force field.

The proposed transmission mechanism allows for mixing three types of mathematical functions to achieve the most efficient representation of the objects. Without the mathematical functions, modifications done in the example will require much larger data files to describe and therefore require longer time to transmit, parse and render.

5.3 General architecture of the proposed framework

There are generally three parts in our collaborative framework: the Language/Script part, the command processing part and the Viewer/Server part.

The Language/Script part could be any interpreted or script languages that can be processed and displayed by the Viewer. This part is closely related to the Viewer/Server since they are responsible for processing and displaying the result of the language/script. The combination can be flexible depending on the complexity of task. Generally, the viewer interprets the language and displays the result, the script is used for executing interactions, and the server is mainly responsible for relaying messages and providing

locking mechanism and consistency control.

The command processing part works as a bridge between the Language/Script and the Viewer/Server. It is not only for relaying messages but also for providing all customized functionalities and operations requested by users. Besides, the command processing part also handles external device input and device-dependent functionalities. The command processing part consists of several basic components as well as optional components for specific jobs. It is also quite easy and fast to build new reliable components to expand the functionality of the current components. Each component in the command processing part is an independent component, and each component can have several totally different implementations. The users have a freedom to choose which one to use by calling the names of the components in the script.

The command processing part also works as an enhancement to the Language/Script which can accomplish complicated tasks. Some of the noticeable advantages for the applications are listed as follows:

1. Customized shapes such as MRI data and PDE-based shapes [127] can be loaded and rendered. The way how to visualize the customized shapes is also flexible, such as displaying Lines/Surfaces/Point Sets/Solid.
2. External functions can be executed inside the Language/Script, which provides an alternative approach of expanding the functionality.
3. External input devices can be incorporated to better support collaboration. The device type will be totally transparent to the Language/Script and the Viewer/Server, which is quite convenient for further expansion.

Based on this architecture, each local operation such as movement and modification

will be converted into a message and sent to the local client viewer. The viewer will then send the message to the server and distribute among all clients in a similar way. Each remote update will also be first received as a message and then executed by the corresponding plug-in components locally. Besides, there are also some other script modules which handle frequently used event/data updates and they do not require message conversion. By incorporating all these generic modules, the overall interactivity is stable and fast, even when external input/output devices such as haptic devices are involved.

The MVC (model-view-controller) is very suitable to illustrate the collaboration architecture shown in Figure 5.2.

Models in Figure 5.2 are shown within the dashed line. They reside in the algorithmic layer and manage both data and behaviour of the collaborative system. The data in this collaborative system are commands which are sent and received as messages. Within the models, these commands are first checked by the prototypes of interfaces to specify which procedures they should go through. After that, the core algorithm in the models will execute different tasks such as parsing, polygonization, rendering, etc. A local message pool is also present in the models to ensure error-free and fast dataflow.

View objects in Figure 5.2 are shown within the bold line. Actually the view objects in the collaborative system reside in the BS Contact viewer, which is located in a visualization layer and handles all user interface elements for visualization. It also receives notifications from the models for faster response and updates the visual content accordingly

Controller objects in Figure 5.2 are shown within the dash-dot-dash line. Actually

the controller objects in the collaborative system are external haptic devices. They receive input from user's interactions and output them to both models and view objects, which are then instructed to translate the user's interactions into actions: the models will process the updated commands, and the viewing objects will update the visual content.

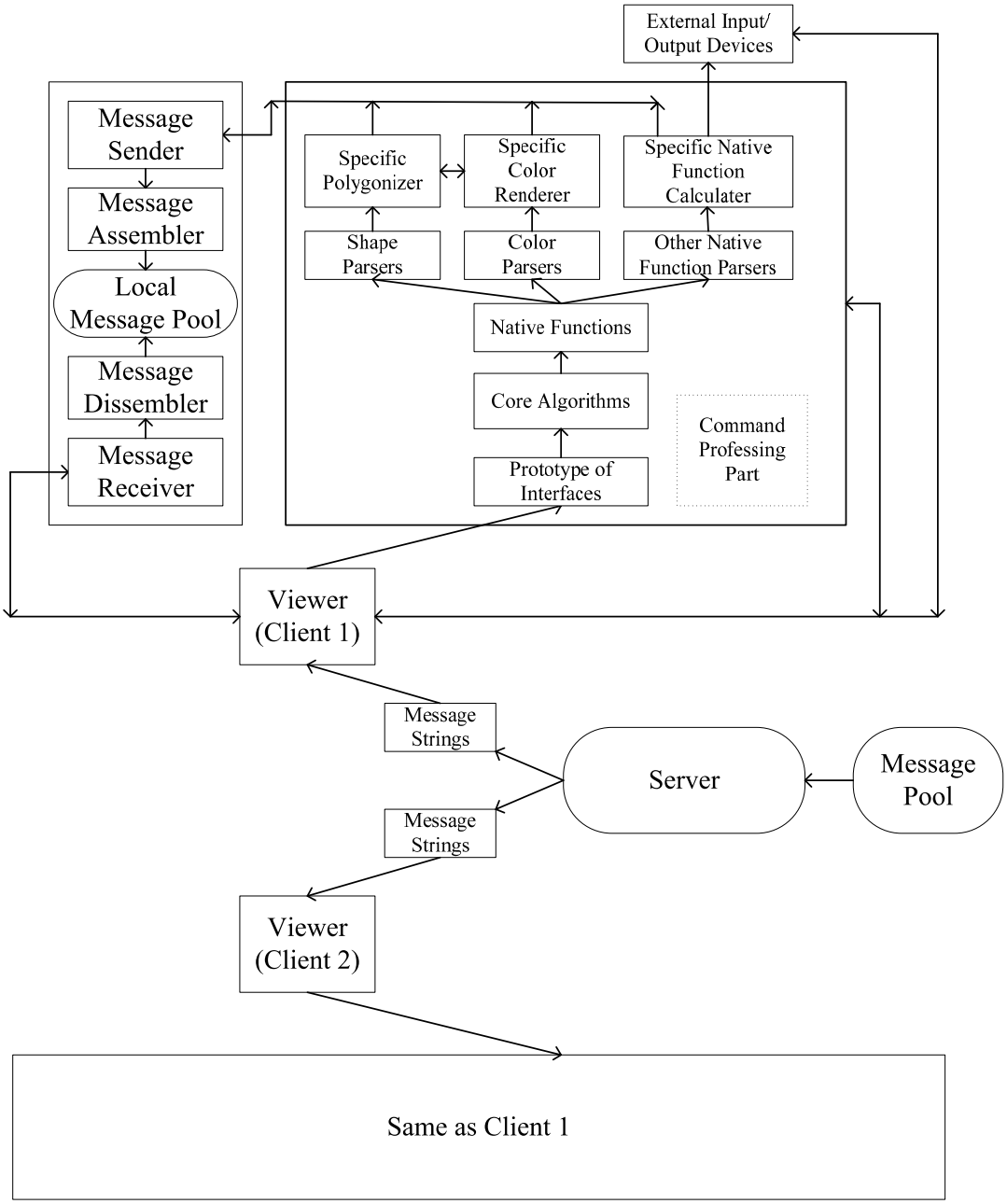


Figure 5.2 Overall architecture of the proposed collaborative framework

5.4 Summary

In this chapter we presented our innovative framework for visual and haptic collaboration. General issues and specific features are discussed in detail to demonstrate the advantages of the proposed framework. Such framework provides necessary components for general networked collaboration and adds new features and organizations to fully support the collaboration between visual and haptic pipelines. The collaborative framework is based on message transmission, which integrates the three types of mathematical functions and procedures into the messages. Any message distributed in the collaborative system can use mixture of the mathematical functions and procedures to accurately and compactly represent complex modelling definitions and operations, which could be problematic to achieve otherwise. Besides, the mathematical functions and procedures are more compact in expressing the modelling definitions and operations than other representations, such as polygonal or voxel-based representations. The incorporation of mathematical functions and procedures has enhanced both the efficiency and extendibility of the framework. This framework also opens research opportunities for collaborative shape modeling with both vision and touch, and implementations based on it will be further demonstrated in the next chapter.

Chapter 6 Implementation in VRML and X3D

In this chapter, we present implementations based on our proposed research novelties. In Section 6.1 we discuss the selection of the implementation platform. In Section 6.2 we outline platform-specific issues. In Section 6.3 we briefly introduce FVRML and FX3D and in Section 6.4 the novel hierarchy of the FShape node is illustrated. In Section 6.5 we demonstrate implementation details of our unified collision detection algorithm. In Section 6.6 we presented two implementations of our collaborative framework based on strong server and thin server architectures. Finally in Section 6.7 we draw conclusions about the implementations.

6.1 Selection of the implementation platform

In order to validate the research niche of the proposed modeling paradigm and collaborative framework, the corresponding software tools and core algorithms should be implemented to support collaborative visual and haptic shape modeling based on commonly available data formats. The implementation should also provide generic and intuitive procedures to create, modify and collaborate with objects based on the unified modeling paradigm.

The implementation should be based on model transmission because it is the most commonly used approach for networked collaboration. As we reviewed previously, there are several existing model transmission standards available such as VRML, X3D, COLLADA, as well as some application specific protocols such as Second Life and

Active Worlds. Since our implementation does not aim to bind to specific applications, we decided to use standard mode transmission protocols.

COLLADA (COLLAborative Design Activity) is a good candidate for our implementation. It is designed as a container for 3D data and mostly used as an exchange format for 3D authoring tools. However, COLLADA is not a standard file format approved by ISO, which restricts its wider usage. Besides and more importantly, COLLADA only defines a static model which reserves property fields without specifying how the properties should be interpreted and represented. Although some efforts have been done to exchange and interpret data using COLLADA in an effective and consistent way, it is still just a data format for exchange and storage and could not be used alone. All interpretation and interactive works are executed by other software including loaders, physics engines and middleware, which load, parse, and execute the data format in their own manners. Such loose relationship between the data format and corresponding behaviours is an obvious obstacle for interactive collaboration over the network, which motivated us to exclude COLLADA from our consideration.

Compared with COLLADA, VRML and X3D have the following advantages:

1. VRML and X3D are international open standard file format approved by the International Standard Organization. For many years they have successfully provided a system for the storage, retrieval and playback of real time graphics content embedded in applications supporting a wide array of domain and user scenarios.
2. VRML and X3D provide necessary programming facilities such as ROUTE and ECMAScript. They could be easily extended and the extensions will still be compatible with the original contents.

3. VRML and X3D could support different user interactions and rendering approaches in various web browsers, which satisfies our requirement for networked visual and haptic collaboration.

Based on the analysis above, we finally decided to use VRML and X3D as our implementation platform.

6.2 Platform-specific issues

Although the implementation platform has been chosen, we could not neglect several unresolved problems of VRML and X3D, which arose during the last few years when attention of researchers and developers has been attracted to 3D internet and virtualization of life:

1. Objects defined in VRML and X3D are only coloured surfaces, lines and points and the transformations are the 3 affine transformations (translation, rotation and scaling). No solid objects and Boolean (set-theoretic) operations can be defined.
2. VRML and X3D are lacking an ability to define physical properties of virtual objects.
3. VRML and X3D are lacking an ability of setting collaborative shared scenes and requires 3rd party software tools to be used for doing this.

To address the first problem, FVRML and FX3D were previously proposed and developed, in which mathematical functions (parametric, implicit and explicit) and procedures are used for defining geometric objects. Besides defining geometry, functions can be used for creating procedural colours, textures and operations on function-defined shapes. The function-based approach with application to VRML and X3D was implemented as their extensions and briefly introduced in Section 6.3.

To address the second problem, in section 6.4 and 6.5, we extended FVRML and FX3D with the definition of physical properties of objects defined by functions.

To address the third problem, in Section 6.6 we presented two implementations based on strong and thin server architectures. Since the performance of web visualization of VRML and X3D scenes largely depends on the size of the models to be transmitted over the network, small function-based models are used to perform collaborative interactive modifications with concurrent synchronous visualization at each client computer with any required level of detail.

In VRML and X3D viewers, polygons, points and lines are used as output primitives. When making extensions of X3D and VRML, there are three ways of how to plug into their visualization pipeline: plug-in customization, External Authoring Interface (EAI) and Scene Access Interface (SAI), and ECMAScript.

Plug-in customization is done by developing a Dynamic Link Library (DLL) using Software Development Kits (SDK) provided by viewer vendors. Therefore, plug-ins are usually implemented in C++. The viewers are only required to download and install the plug-in and declare the respective external prototype. After that, the usage of the extended node is transparent. The major problem of plug-in customization is that it is always browser-dependent. It is impossible to port it from one browser to another even under the same platform.

EAI is used for embedding the viewer into an application or an applet, which is either independent or placed within a Web browser. It can be called through both the COM interface and Java classes. Thus, languages such as C++ and Java can be employed to develop applications or applets through EAI. Besides this, since it is a part of the

X3D/VRML standard, all browsers support it, which ensures better compatibility. However, since it is used for creating external programs, the extensions thus implemented cannot be integrated into the browser.

SAI is the successor and superset of EAI. While EAI only supports to access VRML scenes externally, SAI supports both internal and external scripting interfaces. Besides, SAI also provides better portability among different VRML/X3D implementations.

ECMAScript is the standardized scripting language of JavaScript and VRML script, which supports Java and C++ as well. By writing a specific string in the url field, the browser automatically loads a library file and executes the code provided by the library. This execution method is called native script. The library files for different browsers have only a slight difference, so the source code can be ported with only minor modifications. This method of using X3D/VRML and native scripts can provide the best performance with only a little compromise of the rules. Besides this, any programming language migration can be done quickly since the mechanism does not change.

By comparing these methods, we selected for our implementation ECMAScript and C++ plug-in in order to achieve the best balance between the compliance with the generality and the performance of both VRML and X3D rendering pipelines. It requires from the extending part to create output primitives and add them to the output primitives created by the viewers in their visualization pipelines.

The necessity to plug into the existing visualization pipeline requires that in our visual and haptic rendering pipelines the geometry has to come first followed by the appearance and physics properties which have to be associated with it, as illustrated in Figure 6.1.

Firstly, the visual rendering pipeline is attended. The implicit, explicit and parametric function definitions of the geometry are rendered to create output primitives which are polygons and lines in the VRML/X3D rendering pipeline. If there was any 3D texture defined for the shape by either explicit or parametric displacement of the underlying geometry, it will be taken into account at this stage as well.

Next, the colour is mapped to the vertices of the output primitives. The colour can be defined either parametrically or explicitly. When parametric functions are used, the colour vector is defined as $\mathbf{C} = \mathbf{F}(u,v,w,t)$ where $\mathbf{C} = [\text{r g b}]$ and u,v and w are the Cartesian coordinates of a point in 3D modeling space for which the colour is defined. When explicit functions are used, the value g of the explicit function $g = f(x,y,z,t)$ is mapped to colour vectors $\mathbf{C} = [\text{r g b}]$ by some (e.g. linear) interpolation on the given key function values $\{g_i\}$ and the respective colours $\{\mathbf{C}i\}$. These considerations are applied to different components of the illumination model including diffuse, specular and ambient colours as well as transparency and shininess. The 3D colour thus defined can be then sampled by any geometry (which can be time-dependent) hence making a coloured object. The colour, transparency and shininess can vary across the surface and volume of the object as defined by the respective functions.

The haptic rendering pipeline is engaged when there is a haptic device used with the application. Haptic interaction usually assumes that there is a scene being visualized that serves as a guide for the haptic device. Function parser receives coordinates of the 3D haptic contact point $h = (x,\ y,\ z)$ and three orientation angles of the haptic actuator (*roll, pitch, yaw*). The haptic contact point is then tested whether it collides with any of the objects for which tangible properties are defined. Then, the force vectors to be applied to

the haptic actuator (displacement and torque) are calculated as a result of the combined force created by the surface, density and force field properties definitions.

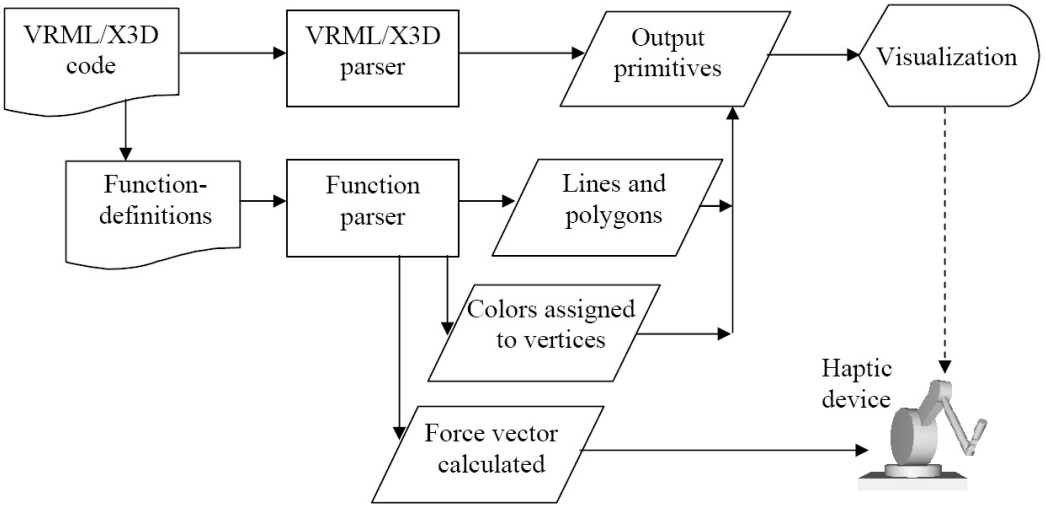


Figure 6.1 Visual and haptic rendering pipelines for function-defined objects in VRML and X3D

6.3 Introduction to FVRML and FX3D

Virtual Reality Modeling Language (VRML) and its successor Extensible 3D (X3D) are international standard file formats commonly used for creating 3D shared virtual worlds on the web, including multimedia and interactive elements. Hybrid function-based shape modeling with application to VRML and X3D was introduced in [128] and further developed as reported in [129, 130, 131, 132]. It allows for defining time-dependent geometric objects, their appearances and transformations by parametric, implicit and explicit functions which can be used concurrently.

The syntax of the extended components follows that of VRML and X3D. The functions are either typed as individual formulas and java-style function scripts or defined as external library functions, for example, 3D reconstruction functions and Partial Differential Equations (PDE).

FVRML/FX3D opens VRML and X3D to practically any type of geometry, 3D colours and geometric textures. It also introduces to VRML and X3D set-theoretical operations (union, intersection, difference), as well as allows for defining any other operations (e.g., geometric and colour metamorphoses). In contrast to the polygon-based models of VRML/X3D, their function-based extension FVRML/FX3D allows for a great reduction of the model size and provides an unlimited level of detail. The defining functions can be functions of time that allows for easy definition of sophisticated animation applied to geometry and appearance on the objects. The function-defined objects can be used together with the standard VRML and X3D objects and appearances, as well as allowing for using the standard object and appearance fields as their parts.

6.4 Hierarchy of FShape with geometry, appearance and physics

Implementing the proposed approach in X3D and its predecessor VRML, new *FShape* and *FTransform* nodes have been added. The *FShape* node, in turn, contains other 10 nodes in the same way how the standard VRML/X3D Shape node is organized, specifically: *FGeometry*, *FAppearance* (with *FMaterial* and *FTexture3D*) and *FPhysics* which contains *FSurface* (with *FFriction, FTension*)*, FDensity,* and *FForce*. These nodes can be used alone as well as together with X3D and VRML standard nodes, as shown in Figure 6.2.

The *FShape* node is a container for the *FGeometry* or any standard VRML/X3D *geometry* node, and the *FAppearance* or the standard *Appearance* node. These nodes define the geometry and the appearance of the shape, respectively.

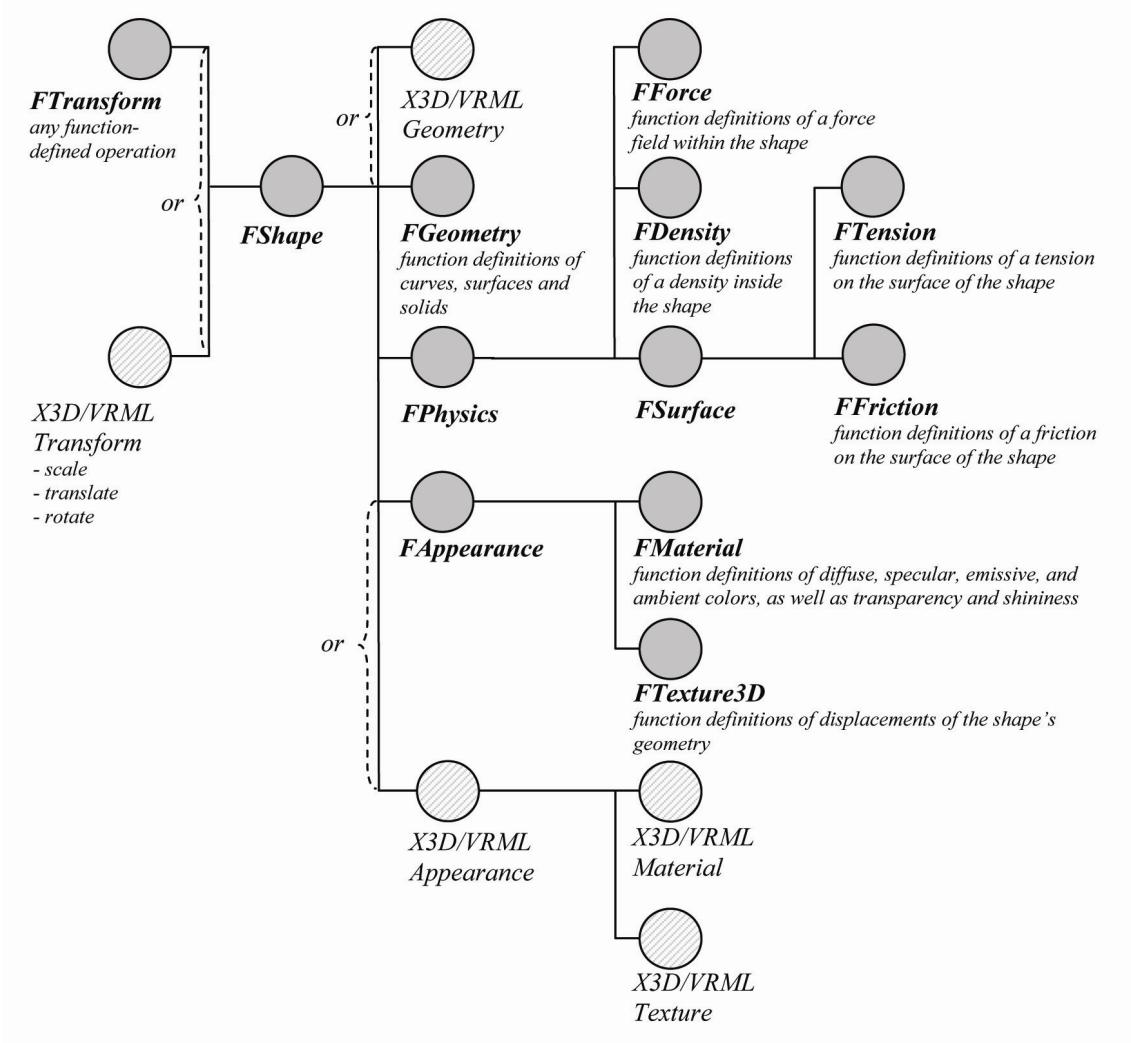


Figure 6.2 Extended function-defined and standard nodes of VRML and X3D

The *FGeometry* node is designed to define a geometry using functions typed straight in the code as individual formulas and java-style function scripts or stored as algorithmic procedures in binary libraries.

The *FAppearance* node may contain the *FMaterial* or the standard VRML/X3D *Material* nodes, as well as the standard colour *Texture* and the function-based *FTexture3D* nodes. In the *FMaterial* node, the components of the illumination model are defined using functions in a way how it can be done in *FGeometry* node. The *FTexture3D* node contains a displacement function or function script for the geometry defined in the *FGeometry* node.

The *FPhysics* node is used for defining physical properties associated with the shape's geometry. The *FPhysics* contains the *FSurface*, *FDensity* and *FForce* nodes.

The *FSurface* node defines surface physical properties. It includes *FFriction* and *FTension* nodes. The *FFriction* node is used for defining friction on the surface which can be examined by moving a haptic device actuator on the surface. The *FTension* node defines the force which has to be applied to the haptic actuator to penetrate the surface of the object. The surface friction and tension are defined by explicit functions of coordinates *x, y, z* and the time *t*. Hence, the surface friction and tension can vary on different parts of the surface.

The *FDensity* node is used for defining material density of the shape. It can be examined by moving a haptic device actuator inside the visible or invisible boundary of the shape. The density is defined by explicit functions of coordinates *x, y, z* and the time *t*, and hence it can vary within the volume of the shape. The surface of the shape can be made non-tangible, while the shape still might have density inside it to define some amorphous or liquid shapes without distinct boundaries.

The *FForce* node is used for defining a force field associated with the geometry of the shape. The force vector at each point of the space is defined by 3 functions of its *x*, *y*, and *z* coordinates and the time *t*. The force can be felt by placing the haptic device actuator into the area where the force field is defined. Moving the actuator within the density and the force fields will produce a combined effect of haptic rendering.

The *FShape* node may be called from the *FTransform* node or from the standard *Transform* node. The *FTransform* node defines affine transformation which can be functions of time, as well as any other operations over the function-defined shapes. These

are predefined set-theoretic (Boolean) intersection, union, difference, as well as any other function-defined set-theoretic and any other operations, e.g. *r*-functions, shape morphing, etc.

6.5 Collision Algorithm implementation

Our collision algorithm is based on a few atomic operations:

- Container identification
- Multiple haptic rays casting
- Haptic rays intersections with haptic containers
- HIP position location
- Surface properties rendering
- Inner properties rendering

Container identification is introduced to handle different types of haptic containers which may or may not coincide with the actual object surface. We implement this operation by assigning unique names to haptic containers and apply one-to-one wrapping between containers and objects. In case the haptic container coincides with the actual object surface, the two surfaces will be identified as one for physical property rendering. If density and force field are defined without an actual object surface, the haptic container surface will be identified as the boundary of the physical properties. In case the haptic container differs from the actual object surface, only the haptic container surface will be identified for physical properties rendering, all other surfaces will be ignored for haptic pipeline and only for visual references.

Multiple haptic rays casting is based on using parametric functions in spherical

coordinates u, v and R:

$$x = R * \cos(u)$$

$$y = R * \sin(u) * \cos(v)$$

$$z = R * \sin(u) * \sin(v)$$

A uniform sampling of u and v provides for an even distribution of the rays in the scene around HIP to the distance R (Figure 6.3).

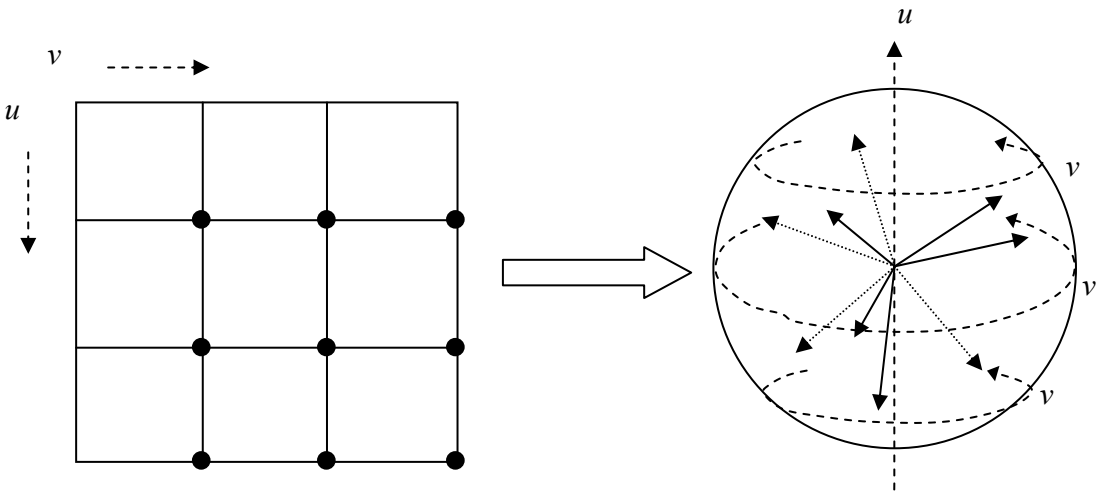


Figure 6.3 Multi-rays generated from equally divided parametric surface

Each haptic ray cast is tested for possible intersections with haptic containers. The containers can be represented either by commonly used polygon mesh based surfaces or defined by mathematic functions (parametric and implicit). In case of polygon meshes a trivial intersection of a parametrically defined ray with the polygons is performed. It can be accelerated by many commonly used ways. In case of parametrically defined surfaces of the containers, the polygon mesh is first calculated and then intersected with the rays. In case of implicitly defined surfaces, the haptic ray intersection can be performed at the level of function definitions. This procedure is similar to that of ray tracing and is performed by iterative binary subdivision steps which eventually result with the coordinates of the intersection point on the surface of the container. Alternatively, the

implicitly defined surface can be also polygonized first however the direct evaluation method has a higher precision, requires less computations and hence provides for a higher performance.

HIP position location assumes understanding of whether the HIP is inside or outside the haptic container, and if inside, how far from the surface it is located. In case of polygonal representation of the containers, the inside/outside predicate is implemented as a dot product $g = \mathbf{R} \cdot \mathbf{N}$ of a haptic ray **R** and a normal vector **N** to the surface at the intersection point (Figure 6.4a). The HIP is inside the container if $g>0$ and outside if $g<0$. The distance to the surface then is calculated as a minimum of distances to polygon planes for the rays cast $D = \frac{|\mathbf{R} \cdot \mathbf{N}|}{\|\mathbf{N}\|}$ (Figure 6.4b).

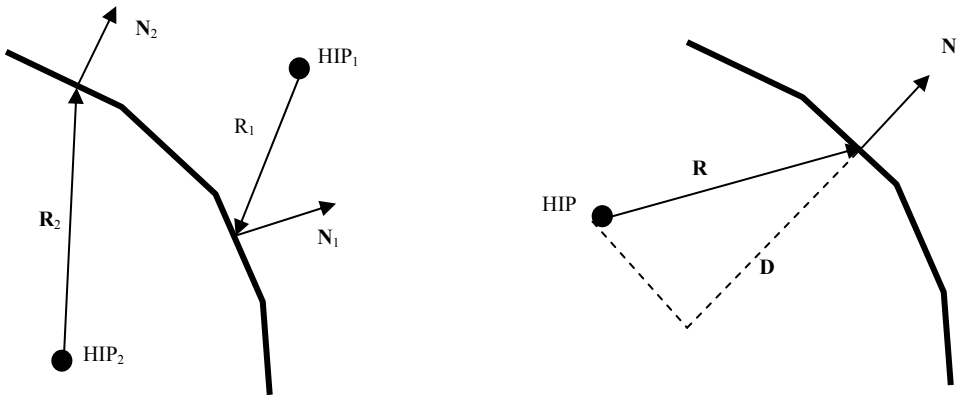


Figure 6.4 HIP location and distance calculation for polygon-based haptic containers

In case of implicitly defined surfaces of the container, the inside/outside predicate is evaluated by analysis of the surface defining function sign which changes on different sides of the surface. Hence, using function definitions $g = f(x, y, z) \geq 0$ for the containers, HIP is inside if $g > 0$ and outside if $g < 0$. Calculating the distance between

HIP and the implicitly defined surfaces we use a gradient to approximate the normal. In order to increase the accuracy of the normal, we use both positive and negative delta and then calculate the average:

$$\mathbf{X}_{normal+} = (\mathbf{F}(x + \Delta X, y, z) - \mathbf{F}(x, y, z)) / \Delta X$$

$$\mathbf{Y}_{normal+} = (\mathbf{F}(x, y + \Delta Y, z) - \mathbf{F}(x, y, z)) / \Delta Y$$

$$\mathbf{Z}_{normal+} = (\mathbf{F}(x, y, z + \Delta Z) - \mathbf{F}(x, y, z)) / \Delta Z$$

$$\mathbf{X}_{normal-} = (\mathbf{F}(x - \Delta X, y, z) - \mathbf{F}(x, y, z)) / \Delta X$$

$$\mathbf{Y}_{normal-} = (\mathbf{F}(x, y - \Delta Y, z) - \mathbf{F}(x, y, z)) / \Delta Y$$

$$\mathbf{Z}_{normal-} = (\mathbf{F}(x, y, z - \Delta Z) - \mathbf{F}(x, y, z)) / \Delta Z$$

$$\mathbf{N}_X = (\mathbf{X}_{normal+} + \mathbf{X}_{normal-}) / 2$$

$$\mathbf{N}_Y = (\mathbf{Y}_{normal+} + \mathbf{Y}_{normal-}) / 2$$

$$\mathbf{N}_Z = (\mathbf{Z}_{normal+} + \mathbf{Z}_{normal-}) / 2$$

The distance is then calculated using the same formula as for the polygonal representation.

Surface properties rendering activates when the HIP is inside the container within its surface zone. Tension force is generated along the normal direction at the surface intersection point while friction force is generate opposite to the HIP movement. In case of the surface zone belonging to a polygonal container, the tension force is rendered as:

$$\mathbf{F}_{Tension} = T_{clipped} * \mathbf{F}_{Device\_Max} * D / L_{SurfaceZone}$$

where $T_{clipped}$ is the clipped tension value of HIP, $\mathbf{F}_{Device\_Max}$ is the maximum output force of the current haptic device, D is the distance between the HIP and the container surface, and $L_{SurfaceZone}$ is the depth of the surface zone at the current position. Clipping is introduced here to cut off the big value and scale it to a reasonable range which the haptic

device can render.

In case of the surface zone belonging to an implicitly defined container, the tension force is rendered as

$$\mathbf{F}_{Tension} = \mathbf{N}_{Approx} * S_{device\_Max} * T_{clipped} - \mathbf{V}_{interpolated}$$

where $\mathbf{N}_{Approx}$ is the approximated normal, S_{device_Max} is the maximum stiffness which the haptic device could provide, $T_{clipped}$ is the clipped tension value of the HIP, and $\mathbf{V}_{interpolated}$ is the interpolated velocity. The velocity is interpolated to neutralize the variations of abrupt force direction change due to approximated normal calculation.

The surface friction is calculated after tension calculation:

$$\mathbf{F}_{Friction} = - Friction_{current\_Pos} * \left|\mathbf{F}_{Tension}\right| * \mathbf{V}_{interpolated}$$

where $\left|\mathbf{F}_{Tension}\right|$ is the magnitude of the tension force and $Friction_{current\_Pos}$ is the friction value of the HIP.

Inner properties rendering activates when the HIP is inside the haptic container but not within the surface zone. For density rendering, the force will generate in the direction opposite to the HIP movement and proportional to the density value. We obtain the velocity of HIP at the haptic frequency and interpolate it as follows:

$$\mathbf{V}_{current\ interpolated} = C_1 * \mathbf{V}_{previous} + C_2 * \mathbf{V}_{current}$$

where $\mathbf{V}_{previous}$ is the velocity at 1/1000 sec before current haptic frame and C_1 and C_2 are interpolation coefficients.

The feedback force is then calculated as follows:

$$\mathbf{F} = \mathbf{V}_{current\ interpolated} * L$$

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

When a force field within a haptic container is defined, the forces there define the

direction of the HIP movement (Figure 6.5a). Since the haptic pipeline executes at 1000Hz, the HIP will be continuously guided to move from one point to another (Figure 6.5b).

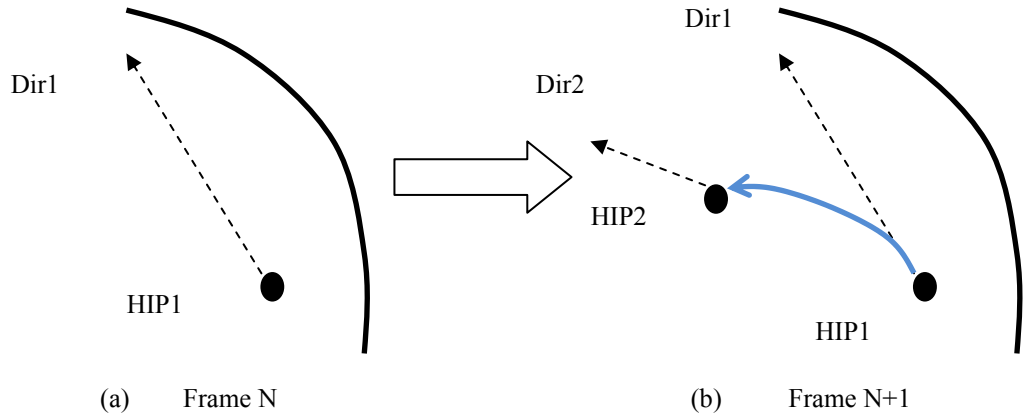
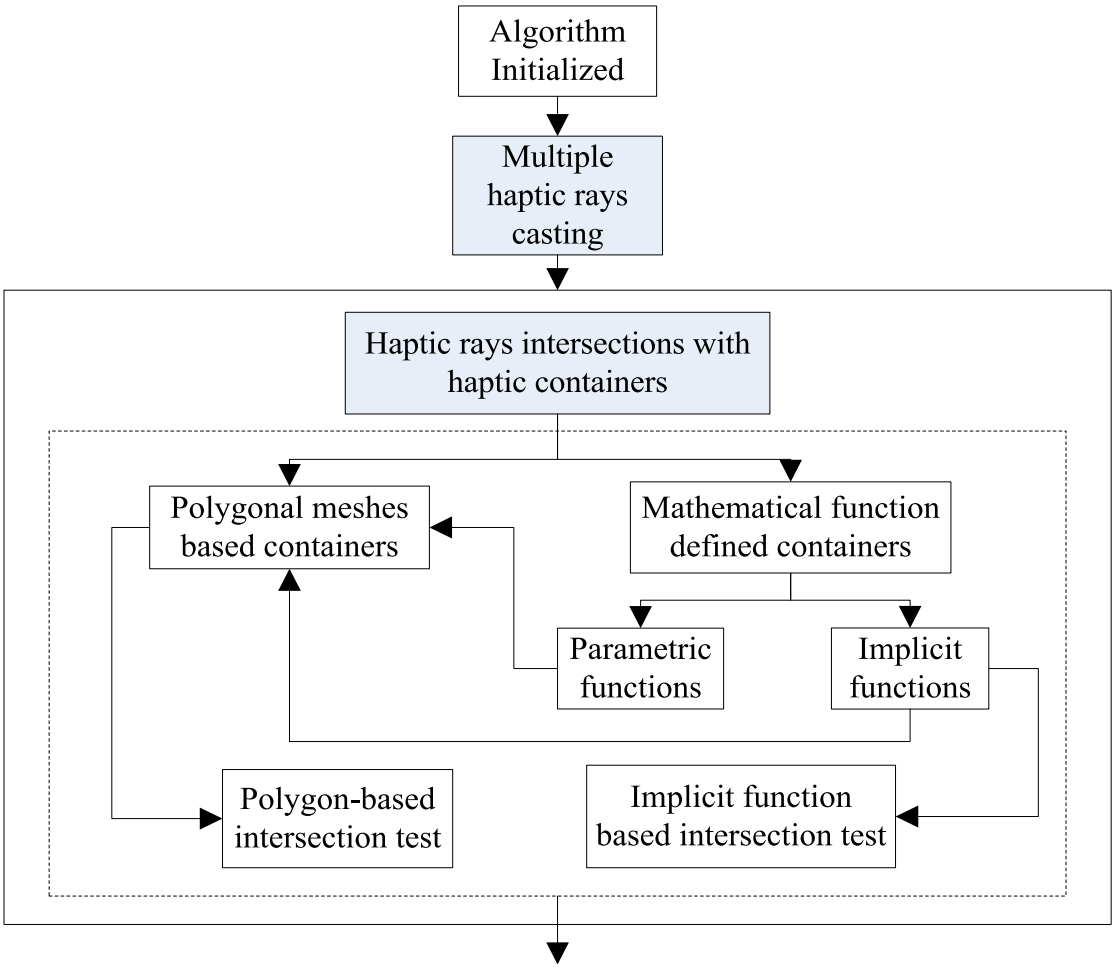


Figure 6.5 Changes of movement direction between haptic frames generate smooth HIP moving path

The algorithm workflow is shown in Figure 6.6.



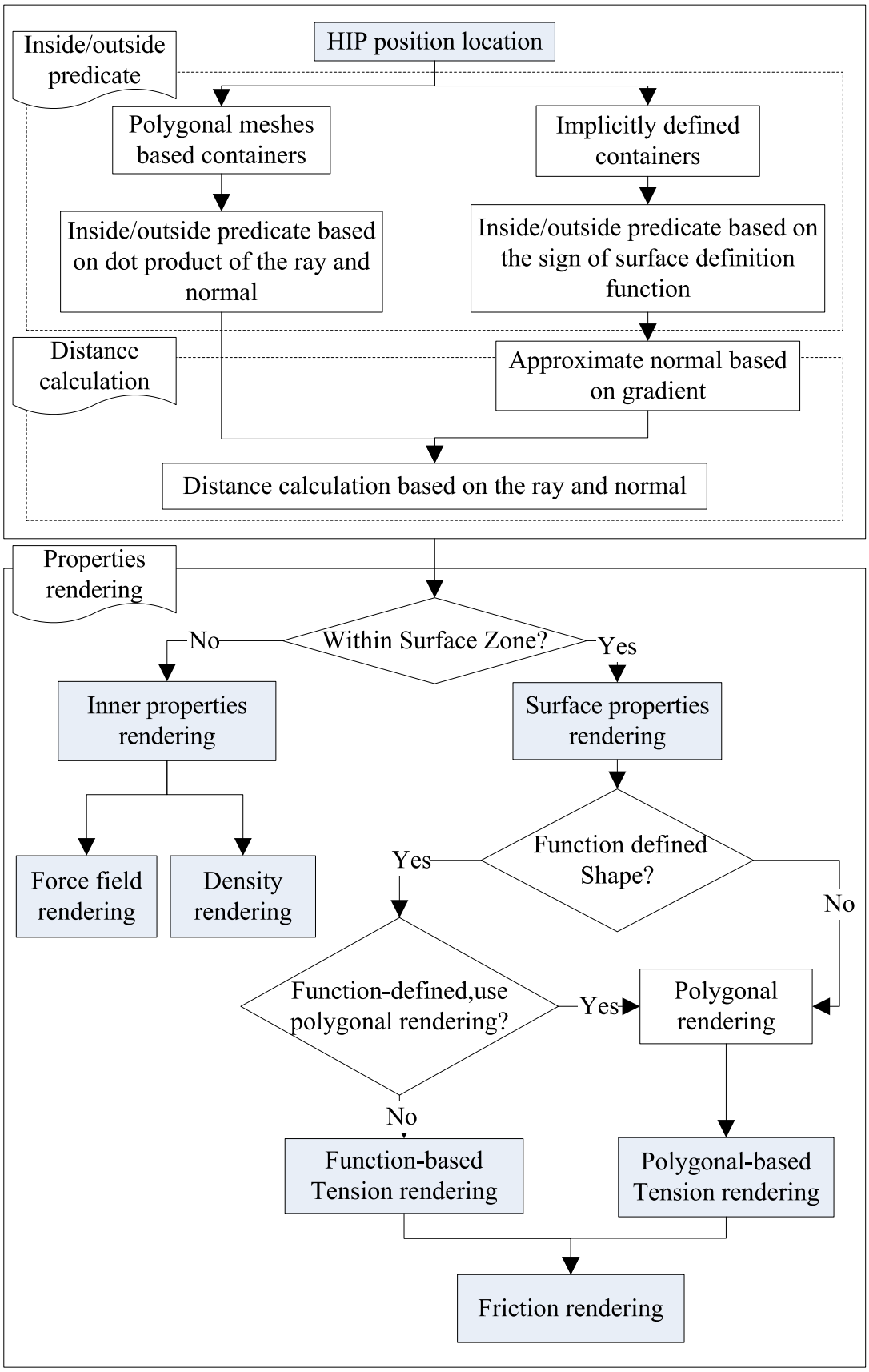


Figure 6.6 Overall algorithm workflow

To implement the proposed approach, we need a generally accepted software platform which requires no direct input of polygons from the user and provides common interfaces for non-specific haptic devices. Besides, as the goal of this PhD project is to build a shared virtual environment which supports both visual and haptic rendering, interaction and collaboration, we need to select a platform which has the potential to support collaboration. Therefore we consider choosing from several available platforms: Xj3D, H3D and BS Contact.

Xj3D is a Java-based VRML/X3D viewer approved by Web3D Consortium. It supports Java3D, OpenGL and some other renderers and works as the test bed for new X3D specifications. Although Xj3D provides almost all features of VRML/X3D, it lacks the ability to natively support shared virtual spaces. Besides, since haptic interaction requires very fast update rate, java-based architecture would be not as efficient as those platforms implemented in compiling languages such as C++. Therefore we have to exclude Xj3D from our consideration.

H3D API is a scene graph based API which can use both C++ and Python to directly load and parse X3D files, and then renders the scene both graphically and haptically. It utilizes OpenGL for graphical rendering and Sensable OpenHaptics toolkit for force rendering. However, H3D also does not have the ability to support shared virtual spaces. Although the H3D viewer could retrieve online data and visualize it locally, it still does not really lead to creating web-based applications

BS Contact from Bitmanagement is widely used for building virtual environments and it supports VRML, X3D and COLLADA. One most important feature of BS Contact for us is that its company provides both client viewer and corresponding server

configurations, which naturally supports collaboration in shared virtual spaces. The BS Contact viewer could also be plugged into major web browsers to further enhance the flexibility for implementing collaboration. Besides, it provides standard COM (Component Object Model) interfaces for supporting various input devices including haptic devices. BS Contact also has some auxiliary methods for interactions, such as ComputeRayHit, which are quite helpful for certain implementations.

Based on the comparison above, we finally selected BS Contact as our test bed.

To be able to support devices from different vendors, we abstracted a group of interfaces that a single-point desktop haptic device could commonly utilize into an "Abstract Device". For different devices from different vendors, the "Abstract Device" maps their specific function calls into the common function calls. If some of the device-specific functions are missing (i.e. some devices may only have 3DOF input), the developer will be able to either ignore these missing input, or design an alternative way to implement them. By using this approach, we have successfully implemented the rotation functions on the 3DOF input Novint Falcon by assigning 3 device buttons as roll, pitch yaw rotations, respectively, and therefore made it a 6DOF input device. Currently, we are supporting desktop haptic devices from SensAble (Omni, Desktop and Premium 6DOF) and Novint Falcon, and more haptic devices will be added into the support list with the expansion of our researches.

Our software is implemented as a plug-in to BS Contact VRML/X3D browser which works with MS Internet Explorer and Mozilla Firefox. A specially crafted string in the url field of the Script node of this browser allows it to automatically load and execute functions from a dynamic link library file. The library is written in C++, which generates

native codes running on the machine. The parser module of the plug-in is implemented using Yet Another Compiler Compiler (YACC) for C. The input file of YACC is a text file which contains the grammar definition of the programming language which is written in a form similar to Backus-Naur Form (BNF). By running the YACC executable, the input file is converted into a C/C++ source file, which parses the grammar of the function definitions.

There are several research novelties in the proposed collision algorithm:

1. The proposed haptic collision algorithm supports concurrent haptic rendering of various haptic effects (feeling of surface tension and friction, as well as interior viscosity and force fields) at any random location of the haptic scene.

   In comparison, other haptic collision algorithms could only provide separate rendering of these haptic effects at predefined, known to the haptic algorithm, locations of the haptic tools. For example, CHAI3D supports multiple material effects such as viscosity, vibration and stiffness. However they implement stiffness as elasticity which prevents from penetrating the surface of the objects to enter the interior part to feel the viscosity. The proposed haptic collision algorithm provides more potential in concurrent definition of several haptic effects on one object.

2. The proposed haptic collision algorithm could define flexible and complex haptic effects to provide better immersion.

   Until now, although several haptic effects have been identified and proposed for haptic rendering, the ways how to define a believable haptic rendering using these effects are yet to be studied. Many haptic research works aim at more precise force calculation and rendering, while it is less discussed how the forces could be defined to

provide the best immersion. For example, in CHAI3D, each haptic effect has only a few parameters exposed to the programmer, which are individually hardcoded in the C++ source file. Each haptic effect can only be rigidly defined and applied onto the whole object, and there is no way to render an object with complex and flexible haptic effects, such as a variable stiffness across the object surface and a variable viscosity inside the object. This limitation greatly affects the content creators to simulate real-life objects using haptics, and the proposed haptic collision algorithm broke such limitation.

3. The proposed haptic collision algorithm could concurrently support various model definitions (polygon-based surfaces, function-based surface models, and volumetric-based solid models) within one scenario.

   In comparison, most haptic collision algorithms could support only one type of the model definitions in a scene, which simplifies the haptic interaction algorithms but greatly limits the abilities of the content creators. The proposed haptic collision algorithm broke such limitation.

4. The proposed haptic collision algorithm is able to identify location of the haptic tool in haptic scenes as well as consistently and seamlessly determine haptic effects when the haptic tool moves in the scenes with objects having different sizes, locations, and mutual penetrations.

   In comparison, other haptic collision algorithms usually use simplified haptic scenes specially created for perfect demonstrations. For example, the sizes of the haptic objects in a scene are always of the same order of magnitude so that the case of concurrent haptic rendering of very large and very small objects is excluded. Also, the

haptic workspace is usually defined with a rigid size and location without abilities of scaling it up or down or moving it forward or backward for reaching objects with different sizes and locations. The initial position of the haptic handle is usually placed outside the objects in the scene, which excludes the situation of being initially inside objects with viscosity or force filed defined. Besides, most haptic collision algorithms seldom consider inclusions or intersections of objects with different haptic properties. For example, when using the CHAI3D SDK to build a scene with one small sphere located completely inside another big sphere, effects from both spheres will be rendered when the handle is inside the small sphere. This may be reasonably explained but could be unrealistic for certain applications, where only properties from the small sphere need to be rendered.

5. The proposed haptic collision algorithm does not require prior knowledge of the haptic scene (primitive-level input such as vertices and normals), everything is done on the fly.

   In comparison, most haptic collision algorithms are heavily dependent on primitive-level input. For certain applications such as plug-ins which could not access primitive-level input, these collision algorithms will not work. Therefore the proposed haptic collision algorithm could be used in more applications compared with other haptic collision algorithms.

In Figure 6.7 a comparison chart between the latest version of HAPI (v1.1.1) and the proposed algorithm is shown to demonstrate the merits of the proposed algorithm:

Features	HAPI	proposed approach using VRML
Collision Handling	Existing algorithms	Original algorithm proposed in the thesis
Force Rendering Algorithm	Existing algorithms	Original algorithm proposed in the thesis
Haptic rendering functions	Rigid architecture	Flexible architecture
Collision Geometry	Supports polygonal models as well as limited available predefined shapes	Supports both polygonal and function-based models.
Surface force rendering	Does not support concurrent rendering of both surface rendering and interior viscosity rendering on a same object	Supports concurrent rendering of both surface rendering and interior viscosity rendering on a same object
Interior Viscosity	Only with rigid parameters	Supports with any explicit function definitions
Interior Force Field	Supports a couple of predefined force effects like (x=1;y=2;z=3;) as shown in their manual	Supports force field with any parametric functions and parametric script definitions (e.g. function parametric_x(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return x/sqrt(x^2+y^2+z^2); } function parametric_y(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return y/sqrt(x^2+y^2+z^2); } function parametric_z(u,v,w) { d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return z/sqrt(x^2+y^2+z^2); })
Prior knowledge of the scene (vertices, normals)	Yes, the algorithms depend on primitive level input	No, object identifications and collision calculations can be done on the fly
Concurrent support of polygonal, function-based and volumetric models in one scene	No	Yes
Surface force rendering using functions	Position function effect (Supports analytically-defined implicit functions but does not support function script definitions)	Supports analytical-defined implicit function, explicit function and function script definition

Dynamic haptic effects (Time support)	Supports with parameter t only. Concurrent definition with both coordinates (x,y,z) and time (t) is not supported	Supports any function-based definition with x,y,z, t or u,v,w,t
Ease of use	Only for programmers who know C++ very well	Mid-level computer users who can write simple VRML/X3D code and ECMAScript/Java Script
Time required to build new scenes:	Each scene needs to be written from scratch with pure C++ code	Existing VRML/X3D scenes can be easily downloaded from the web. Minimal modifications are required to make them work with haptics
Complex scene with various objects & haptic effects and mutual penetrations.	Not supported	Supported
Enable users to adjust to the best way to explore the scene and each object	Not supported, all scenes are specifically defined to fit the window, no very large or very little objects, no very far or very near objects	Could use adjustable haptic zone to navigate to any location of a very large or small scene and examine the interesting part with any level of precision
Web-enabled	Not supported. Each demo is a compiled executable which runs on local computer.	Supported. Collaborative visual and haptic interaction in a shared scene could be established easy and fast

Figure 6.7 Comparison chart between HAPI and the proposed algorithm

6.6 BS Collaborate server implementation

To develop a VRML/X3D-based interactive application which can be called from a web browser, a so-called native script execution method has to be used. By writing a specific string in the *url* field, the browser automatically loads a library file and executes the code compiled from Java or C++ languages. The library files for different browsers have only a slight difference, so the source code can be ported to them with only minor modifications.

Hence, there can be 3 parts in such collaborative application: html-code, VRML/X3D code with scripts, and binary libraries. All these components can be stored on the server and client machines. These three parts should be able to exchange data during the run-time of the application. This can be done in a few ways briefly outlined

below.

1. The data between the html code and VRML/X3D with scripts is exchanged both ways using the special functions of the ECMAScript language.
2. The data within VRML/X3D with scripts is exchanged through the standard mechanism of data exchange such as event variables and exposed fields. A significant portion of the interactive software can be written using scripts and SDK functions provided by the respective VRML/X3D viewer, however this way is normally limited in how the applications GUI can be implemented and restricts computational complexity to the limits of scripts which are abridged ECMAScript codes.
3. The data between the VRML/X3D with scripts and the Java or C++ functions stored in the DLL libraries can be exchanged both ways. The functions in the binary libraries can take most of the computational load of the interactive application while still using the graphics pipeline implemented at the VRML/X3D level. If the application is designed in such a way that no further extensions are expected, all the data flows can be planned in advance and implemented through the function parameters and function values. However, if new components of the application can be developed in the future, there is a challenging problem of exchanging the data between the binary functions stored in different DLLs, which have been independently compiled without making references to each other. This can be done by setting up an exchange protocol in the RAM and running a special daemon process that should start before any component of the application is loaded and continue running while the application is being executed. When any of the components needs to access the data, it will query the daemon by providing certain flags. The daemon will then check for the flags and return a pointer

to the application component. After that, the component will use the returned pointer as a local pointer for accessing the data it needs in the shared memory.

The described data flow exchange is illustrated in Figure 6.8.

To support collaboration in VRML/X3D applications, we proposed two frameworks based on the concepts of the strong and thin servers. The strong server controls all the issues concerning the synchronous visualization of shared objects on each client computer including storage of the parameters of the objects being shared. The thin server only provides locking and relaying messages between the client computers while the shared objects are implemented by exchanging models between the applications on the client computers.

The synchronization between haptic database and visual database is done by the local plug-in. It adopts asynchronous scheduling mechanism to retrieve and assign values between visual and haptic pipelines. As normal visual update rate is lower than the haptic update rate, the two pipelines runs at their own speeds and exchange data when necessary. Since each client has installed the plug-in, local synchronization between the two pipelines could be achieved. In the collaborative environment, the server does not care which data are for the haptic pipeline and which data are for the visual pipeline. The server just relays the data to the receiving clients at its fastest possible speed. The receiving clients will get all the data they need and send them to local visual and haptic pipeline respectively for corresponding updates.

The haptic update rate is at 1KHZ which is faster than visual update rate that varies from 24HZ to 60HZ. This leads to massive generation of shared events from haptic pipeline which were transmitted across the collaborative platform. The collaborative

server maintains a message pool which size could increase upon client request. Therefore, all generated shared events will be stored in the server message pool and relayed to their destinations at the fastest possible speed. Since we adopt mathematical functions and procedures to represent the transmitted messages, they are compact enough to be transmitted interactively. The collaborative server adopts UDP protocol to deal with bandwidth and latency problem. Besides, an optimized message resending mechanism based on prefix and suffix of messages is also used to handle possible errors during transmission. Therefore, both speed and correctness of the transmission could be ensured.

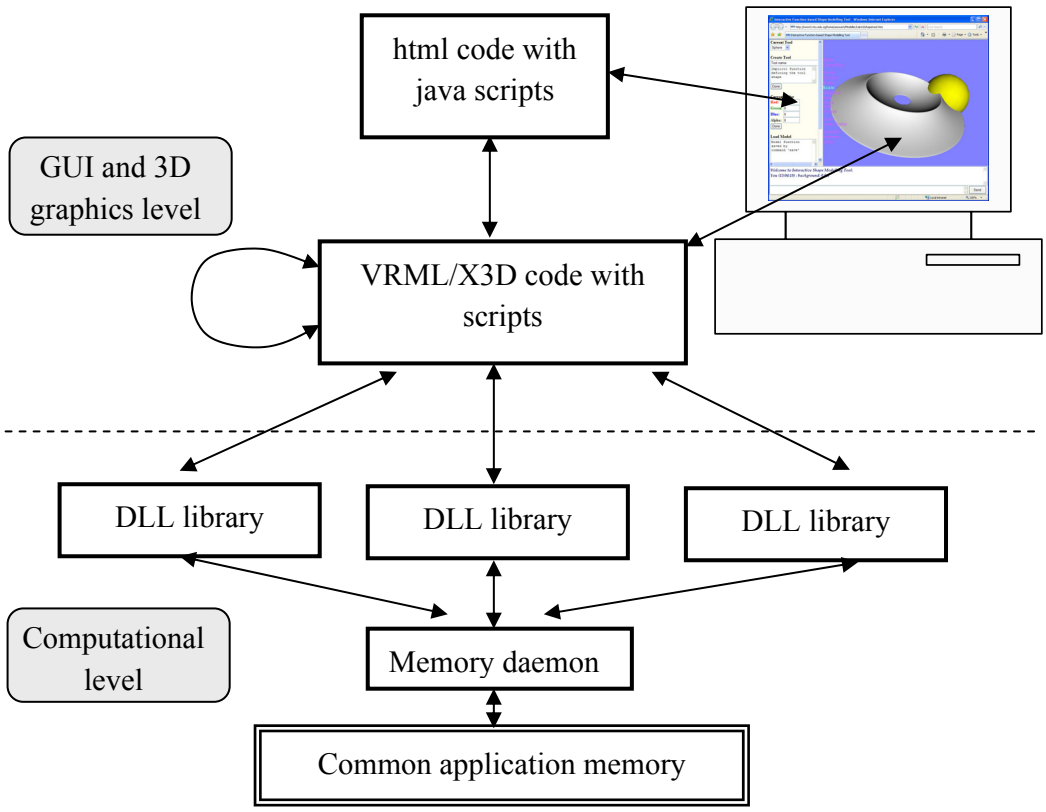


Figure 6.8 Data flows in VRML/X3D-based interactive web-enabled applications

Since there is no native support for collaboration in VRML and X3D, a third party collaborative platform is required for implementation. We have no intension to develop a new platform ourselves; therefore we need to adopt one.

There are several platforms developed for supporting collaboration in X3D and

VRML scenes.

DeepMatrix [133] is a popular open-source multi-user 3D communication platform that features chat, shared objects, and shared events. It is written in Java consisting of one server application and one client applet that works with different web3D plug-ins. DeepMatrix can use either existing VRML browser as the client via the Java EAI, or its own VRML client written in Java. The DeepMatrix's functionality includes text chatting facilities and shared events. However, the shared events in DeepMatrix do not provide any support for a server side locking mechanism, which is very important for our proposed framework. DeepMatrix also uses a rigid GUI which follows Blaxxun style, which imposes serious restrictions for our design and implementation, especially when both visual and haptic pipelines are involved. Besides, the java-based architecture is less efficient than C++ based architecture, which could have negative impact on the update rate of haptic pipeline.

Blaxxun Communication server [134] was developed to support shared VRML. To a certain extent it could support X3D worlds if X3D files are called from VRML root file and used with BS Contact VRML/X3D viewer. It was capable of providing collaborations using client-server mode, however the update rate of the platform was hardcoded at 3 times per second, which is rather at the low end to implement real-time interactions. Also, though our research group had a full license of the server, the software is no longer offered and supported on a complementary web access basis as it was done in the past.

ABNet software [135] is a java-based tool for turning single-user X3D scene into a multiuser environment. It is based on client-server model and it can support asynchronous shared environments using XML messages sent by TCP/IP. It provides an experience

quite similar to Blaxxun Communication server, while both the client and server are free for non-commercial and educational purposes. However, the ABNet software is targeted at light-weight VRML/X3D shared environments where only gestures and text chatting are involved. The lack of server-side locking, shared events and shared objects makes it not suitable for our implementation.

Planet 9 GeoFeeder Server [136] is a multiuser server which mainly focuses on geo-visualization applications. RayGun is the corresponding client 3D viewer which supports X3D and provides tracking, navigation and social networking. However, since GeoFeeder Server is developed for geo-visualization services such as GPS tracking and data metering, it is not suitable for general purpose collaborative applications.

The Octaga Collaboration server [137] is a newly developed collaborative server which is upgraded from the Octaga MPEG4-based MU server. It supports core and interactive profiles of X3D and provides real-time interactions by the recently introduced *Connection* and *NetworkSensor* nodes. Collaboration can be implemented by using both the server and client modes of the Octaga Player—the corresponding client for viewing X3D content either as a standalone application or as a plug-in to mainstream Internet browsers. However, their Connection and *NetworkSensor* nodes only support the UDP protocol. TCP based data transmission is not implemented. Therefore it is not suitable for our implementation, especially when reliable transmissions are required.

The BS Collaborate server [138] is a networked communication platform which supports VRML/X3D multi-user interactions in collaborative shared environments such as text chatting, text-to-speech, server side computation and client identification mechanism. BS Collaborate works together with the BS Contact VRML/X3D client. The

pilot version of BS Collaborate was developed for setting up simple shared environments and was lacking server-side locking and shared objects, however their developers are active in incorporating new features and pluggable modules into it as well as providing configurable versions for independent users, which makes it quite suitable as a test bed for our own implementation.

Based on the comparison of the above tools, we have chosen BS Collaborate server as an implementation platform. We started cooperating with Bitmanagement Software GmbH in 2008 after we had tested the pilot version. The cooperation results in a newer version of BS Collaborate server and BS Contact VRML/X3D viewer which are specifically implemented for our requirements. The version of BS Collaborate server we use supports information transmission, shared events, shared objects and server side locking. Compared to other collaborative platforms, it does not impose restrictions on the shared scene, such as fixed file framework and window layout, and leaves the developers very much in control of most of the collaborative issues. However this version of BS Collaborate has never been published on their website. These software tools work as a client-server pair to support visual and haptic collaboration in shared virtual spaces defined by X3D and VRML and, optionally, by their extensions such as the function-based extension which we use for defining physical properties of the virtual objects.

The version of BS Collaborate server we use is an executable file with several corresponding configuration files. It binds to a specific computer name and IP address and monitors possible connections. The rest of the collaborative system is mostly on the client side, which is written in VRML and X3D. The VRML/X3D file contains a node

which specifies how to connect to the BS Collaborate server. Bitmanagement Company provided us the BS Collaborate server as well as a VRML template and several Prototypes to illustrate how the nodes and fields could be utilized. The template is mainly about the EventStreamSensor which transmits shared objects and events in the shared environment. The Prototypes are used as demos to illustrate server side locking, shared events and consistency control. Based on what they provided, the thesis author proposed the concept of strong server and thin server architecture using BS Collaborate, two modes for communication using text strings, comprehensive routines to convert all modifications of visual and haptic properties into string messages, and all other ideas related to haptic interaction. After that, the thesis author implemented the overall collaborative modelling logic, including message assembler and dissembler, native function parser, user interface, and all other application logic related to haptics.

Besides BS Collaborate Server, our collaborative platform also uses an HTML server (Apache). The two servers work together to provide the functionalities of the framework. The HTML server is based on TCP/IP protocol, which also connects to a database for user registration and scene retrieval. The BS Collaborate server provides the functionalities of information transmission, shared objects, shared events, synchronization and consistency control, and it uses UDP/IP protocol. The two servers use different network protocols and they are responsible for separate issues.

6.6.1 Thin server implementation details

This collaboration framework (Figure 6.9) expects very few server functions to be used for maintaining collaboration whilst relying on the clients to implement the rest of them.

The server only provides locking mechanism and information transmission between the clients. The client-based software is responsible for sending to the clients through the server all the modified object models whenever any of such modifications occur. This approach makes the application software independent of the collaboration server used and allows the developers to easily implement the maximum sharing ability by exposing all shared fields to run-time changes: all properties of shared objects and tools can be dynamically changed and synchronized, including geometry, appearance, and physical properties. This method is particularly efficient when relatively small function-defined (procedural) models are used for defining shared virtual objects and their properties however we used it with standard VRML/X3D shapes as well.

All collaborative modifications are controlled by the clients. For each of the properties that are going to be shared, a special routing script has to be set up for monitoring events and recomposing output. When one client initiates an action, it will first request the lock from the server. After that, the interactions such as geometry modification and colour modification will be received by the local routing script, and a corresponding output based on the client's status (e.g. object's location, orientation, etc.) will be composed. The interaction will cause the server to relay the updates that will be finally received and executed by all the clients. The server is not responsible for storing scene status. Instead, one of the users in the scene must be responsible for temporarily storing current status of the scene. When all the clients leave the scene, its status will be lost unless it is somehow stored manually before (e.g. 3D snapshot of the scene).

The main events in the collaborative framework are:

- New client joins the modeling session;

- The model is modified;
- New model is loaded;
- The model has to be saved or exported for further use.

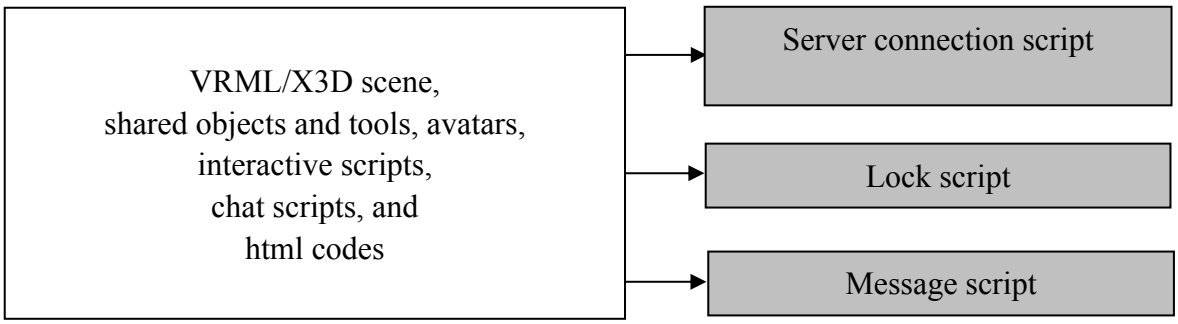


Figure 6.9 Setting shared and collaborative features of the scene with the thin server used

The details of the events implementation follow.

When a new client joins the current collaborative session, the user has to see the scene in its current status with all the modifications that could have been made by other users before the new one joined them. It means the most recently updated model of the scene has to be forwarded to this client. When a new client has been initialized, it first detects whether the joining session is a new session or an existing session. This is done by inspecting the *Initialization lock* implemented by the *EventStreamSensor* node. This lock is obtained by the primary client who joined the session first. This client will be responsible for sending the current modeling object to new clients joining the design session. If the client finds that the *Initialization lock* is available, the session is declared as a new session. For a new session, some original or void scene has to be loaded and visualized. If the client realizes that the *Initialization lock* has already been acquired, the client sends a message to request the current scene. When the primary client left the session, the *Initialization lock* will assign one of the available clients to be the primary

client.

When the scene is modified by any of the clients, the modifications must propagate to all other clients and update the scene as they see it on their computers. The scene can be defined by the standard or extended VRML/X3D. It is only essential that the scene description has to be done in ASCII form to be transmitted as messages and that the scene browsers at the client computers should be able to render the received scene descriptions. When an object modification is done by any of the clients, the *Modification lock* has to be checked. If any other client is modifying the shape at the same time, the modification to the current model will be ignored. Otherwise, the modification is converted into a message and sent to the server which will broadcast it to all the clients participating in the collaborative session. The *Modification lock* will be released by the client after the modification message is successfully sent.

When a new scene is loaded or reset by any of the clients, the new scene has to be delivered to all the participants. Since the design session is web-based, the scene model can be entered through an HTML text box. Then, it will be sent as a message by a certain java script to all the clients participating in the design session.

Finally, the users should be able to save the design at any time on their client computers since the server is not responsible for storing such information as in the case of the strong server. The source code of the scene is assembled as strings in X3D, VRML or any other ASCII formats which the application may require. After this, the strings are printed in the console pane of the VRML/X3D browser, since neither VRML/X3D script nor java script in HTML can access local hard drives due to security reasons. The code can be then manually copied from the console panel to a file, which the client will open

on the hard drive.

The messages are transmitted over the network by the pair of VRML/X3D prototypes which are responsible for sending messages from the user and for receiving messages from any current users, respectively. These prototypes naturally integrate with the server-side locking mechanism, which means that if there are messages transmitting, all other messages from other clients will automatically be ignored. The length of the message size is configurable, with a default size of 4096 bytes. Depending on the estimation of the message length, longer or shorter chunk sizes can be set to find the balance between the transmitting efficiency and the Internet bandwidth. When a new message arrives, the message is processed by the local script. When a complete message has been received, different methods are called according to the message type. If the message contains the whole scene, the current modeling scene is replaced. This is used for loading a new scene or initializing the current modeling scene when a new client joins the session. If the message contains modifications, the current modeling scene is modified according to the modifications. If the message requests the current modeling scene, only the primary client will send back the whole current scene. Since the presence of CRLF characters is essential to make the text messages editable when we need to save the scene source code, we replace CRLF with special characters and restore them after receiving the message. In order to solve the partial transmission problem, we added a prefix and suffix to each message transmitted over the Internet. When the prefix is found in the received message, the modeling tool begins to accumulate the received data until the suffix can be found. If error happens during message transmission, the receiver will identify the broken message blocks and ask the sender to resend them. By doing this, we can successfully

receive the messages.

6.6.2 Strong server implementation details

When this method is used, any VRML/X3D scene can be made a shared collaborative scene by adding to the scene root file a few script modules that set sharing and collaborative features and parameters (Figure 6.10). This approach allows the developers to control all aspects of the collaborative application while using all of the server features for supporting the collaboration.

BS collaborate allows for defining the viewer's avatar, text and text-to-speech (TTS) chat, as well as shared objects and tools associated with various interactive and haptic devices. Physics properties can be optionally added to shared objects and tools by using the function-based extension of X3D and VRML. More advance features allowing for user management can be implemented by using 3rd party web-servers, script generators and databases. Each of the shaded modules in Figure 6.10 are scripts and prototypes which only require to add names (URLs) of the VRML/X3D files defining the actual avatars, prototypes of the shared objects and tools and their initial positions and orientations.

The core part of the server implementation is the *EventStreamSensor*, which is a specific data stream for transmitting certain objects and events in the shared environment between a network stream and the X3D event graph. All the values that are transmitted through the *EventStreamSensor* and stored on the server resemble the scene state. By incorporating this mechanism, any field values in an X3D/VRML scene can send and receive updates via a TCP/UDP socket or HTTP server. To increase the flexibility in

application design without increasing the complexity of the implementation, it is possible to have multiple *EventStreamSensor* nodes in the scene that are assigned the same stream name, which allows for sending events from one part of the scene to another. Besides, the associated stream name can be changed dynamically. This makes the *EventStreamSensor* node disconnect from one stream and connect to another. It allows a script to purposefully select a stream and set values there. We use an implementation of this mechanism from Bitmanagement Software GmbH to initialize a shared object before we create it.

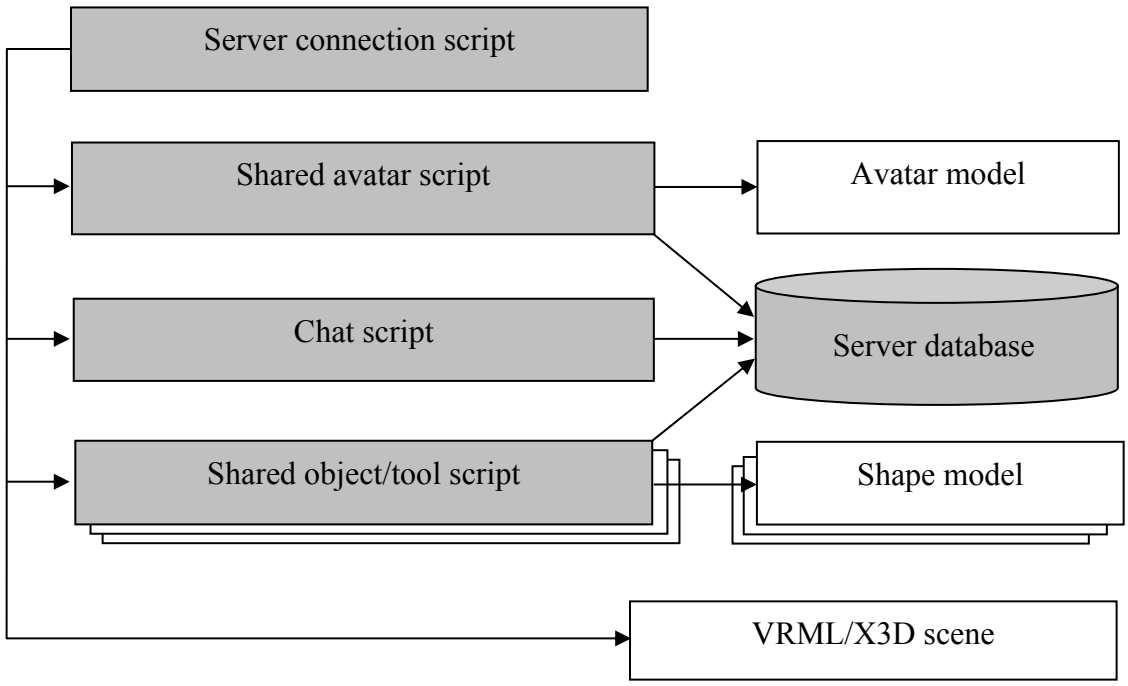


Figure 6.10 Setting shared and collaborative features of the scene with the strong server used

Shared objects and tools have to be defined as their shape prototypes. Each prototype may have several instances available concurrently in the shared environment. In the shared objects/tools prototypes, any number of fields can be exposed to the server and updated by various user interactions. These exposed fields can be position, orientation, geometry, appearance, physical properties, or even the whole shape node. During the collaboration run time, shared events will be generated by different user interactions, such

as movements, clicks or modification commands. These shared events are then received and interpreted as scripts which are sent into the shared object/tool module prototypes, triggering the corresponding field changes. By doing this, all instances of a certain shared object/tool prototype will be updated simultaneously. Since this procedure is broadcast by the server and identically executed on all the clients, they all will be updated adequately. Besides, since only the update events are transmitted, the server will not be overloaded. We have tested the maximum possible shared events update rate by using haptic device interval as the trigger (1000HZ) which is beyond normal speed of user interaction. The result was quite acceptable for making interactive shared collaboration (less than a 1 sec delay).

To ensure that all new clients display the current state of the scene when they connect, the server makes a back up of all the shared fields in a database to send them to the new clients when they log in. Even if there are no users in the scene, the scene parameters are still kept for further use. There is also a possibility that the application code updates a value too frequently. In that case the client is allowed to drop or consolidate events if too many events are sent for the bandwidth available and if the reduction does not change semantics. Besides sharing the scene state, clients also need to communicate with each other for performing a consistent presentation. However, this is not a state that should be stored, and new clients joining later should not receive the event.

Server-side locking is another important feature of the server. To allow for a server side locking mechanism, special fields have been added into the *EventStreamSensor*, which allow for defining different modes of server-side locking (strong, weak), as well as

information of lock owner and lock states. Every time the locking state of a stream changes, the respective field of all *EventStreamSensors* associated with that stream emits information about the client who owns the lock now. This information includes the client login name.

6.7 Summary

In this chapter we presented our implementations to validate the research niche of the proposed modeling paradigm and collaborative framework. VRML and X3D are selected as the implementation platform after serious consideration and comparison, platform-specific issues are also discussed. We implemented a haptic plug-in to Bitmanagement BS contact VRML/X3D browser. We described the architecture of the plug-in and the branches of the core collision detection algorithm. Various physical properties can be concurrently rendered by touching geometric container of the objects, which may or may not coincide with the actual geometric surface. Still staying within the VRML/X3D rendering pipeline, we are able to perform concurrent haptic rendering on VRML and X3D scenes mixed defined by functions and polygons, with different desktop haptic devices from different vendors. Jointly with our research partner Bitmanagement Software GmbH, we have also developed two implementation frameworks based on the strong and thin server concepts, which can be used for making shared collaborative environments with both standard X3D and VRML as well as their extensions. In the next chapter several comprehensive examples and practical collaborative applications will be presented to verify the development and practical value of these implementations.

Chapter 7 Application Examples

In this chapter several comprehensive examples and practical collaborative applications are illustrated to verify the development and practical value of the ideas proposed in the project. In Section 7.1 several haptic modeling examples are discussed in detail to present advantages of the unified modeling paradigm. Demonstrations of practical collaborative applications are presented in Section 7.2 to testify the usability of the visual and haptic collaborative framework. In Section 7.3 we draw conclusion about the practical value of our research niche and implementations.

7.1 Simple haptic modeling examples

We are going to discuss several haptic modeling examples in this section. Mathematical functions and procedures are used in the examples to illustrate the compactness and extendibility of our research niches. To ease reading and save space, the codes of the examples are written using VRML-style encoding however XML encoding for X3D can be used as well.

In Figure 7.1, an example of modeling a tangible "watermelon" is given. The geometry, appearance and physical properties of this object are defined with explicit FRep functions. The geometry is defined as a sphere with a piece cut away from it by a semi-infinite solid object defined by 3 planes:

$$func = \left(1 - x^2 - y^2 - z^2 \right) \ \& \ \left(-\left((z - x) \& y \& z \right) \right) \ \geq \ 0$$

where the set-theoretic intersection is denoted as &. The Set-Theoretic Operations (STO) are implemented in the extension both by min/max and *r*-functions. It is up to the user to

decide which continuity of the resulting function is required to select the (set theoretic operations) implementation to be used.

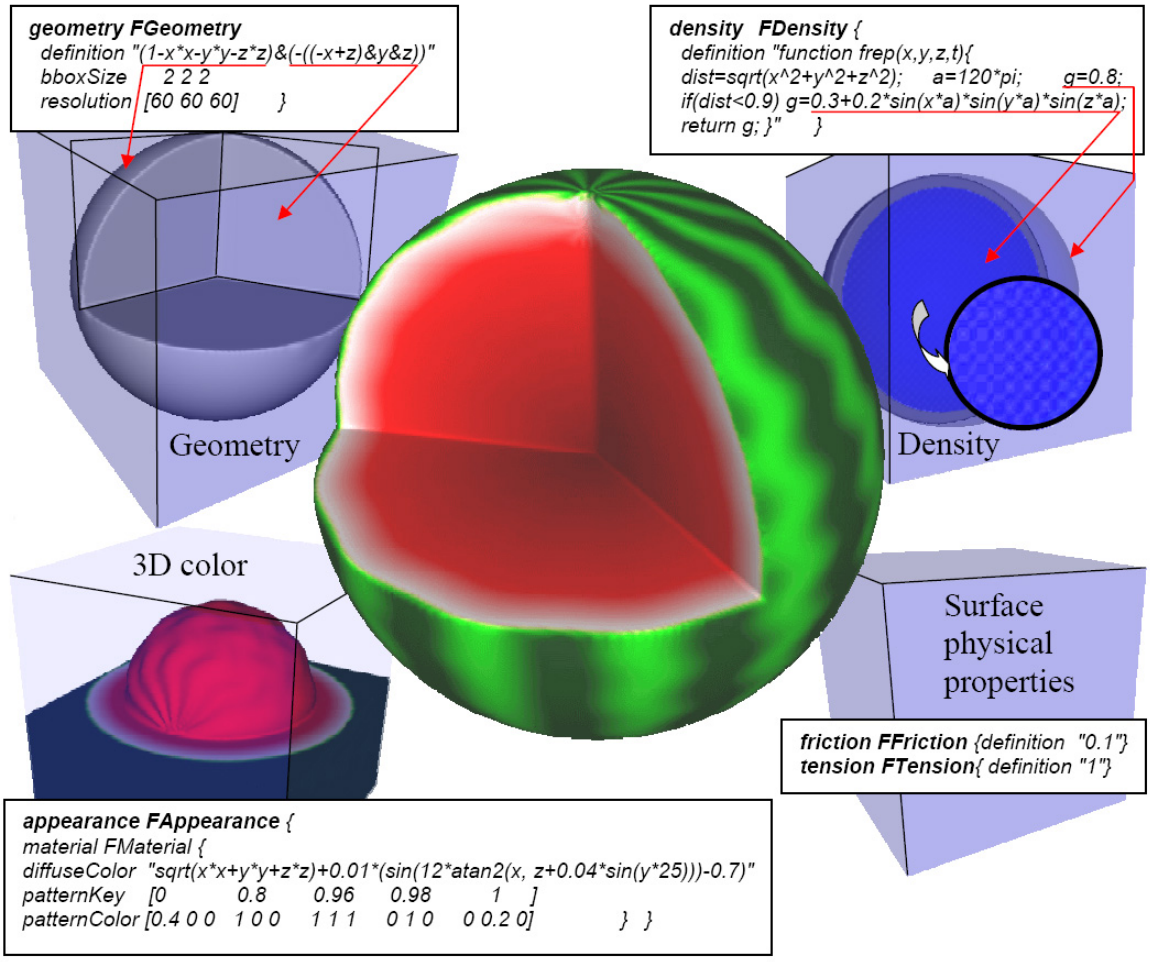


Figure 7.1 Modeling "Watermelon" by assembling the object from various properties

The colour is then represented by a 3D colour field defined in the same geometric coordinate space. The function of this field is defined as a distance from the origin

$$dist = \sqrt{x^2 + y^2 + z^2}$$

with distortions defined by a specially designed noise function

$$noise = 0.01\Big(sin\Big(12atan\Big(x, z + 0.04sin\big(25y\big)\Big)\Big) - 0.7\Big)$$

The function values are then linearly mapped to the colour values according to a designed colour map. The uneven shape of the colour iso-surfaces was used for making

green patterns on the surface of the “watermelon”. This was achieved by mapping the respective function values from 0.98 to 1 to RGB values of colours ranging from *[0 1 0]* to *[0 0.2 0]*. The colours inside the watermelon are also created by linear mapping of the respective colour function values to different grades from yellow, through white, and finally to red colours.

The density inside the watermelon is defined as a distance function script with a value 0.8 near the surface and a variable value *g* inside it:

$$g = 0.3 + 0.2\sin(120\pi x)\sin(120\pi y)\sin(120\pi z)$$

The density values are displayed with different colours and intensities. The zoomed section of the inner part shows how the density changes to simulate a crunchy body of the watermelon. The surface friction and tension are constant 0.1 and 1, respectively. The whole FShape object definition code is built by putting together the four fragments of the code given in Figure 7.1.

In Figure 7.2 we make a scene from standard and function-defined shapes. The fan is made from standard polygonal VRML objects. It is defined as a touch sensor so that when it is clicked its blades will start rotating. The surfaces of the standard objects—fan, cabinet and carpet—are made tangible by grouping them with a function defined object which has only a tangible surface property defined. An air-flow is defined functionally within a transparent geometric cone which is attached to the fan blades, as displayed in the figure. The surface of the cone, which is defined by parametric functions, is used for triggering collision detection. When the fan starts blowing, the force field will be activated as well, and it can be felt with a haptic device when its virtual contact point is placed inside the invisible cone. In this example, we also illustrate how different

properties can be modelled in different coordinate domains and then mapped into one object. Hence, the force field is originally defined in a unit bounding box which is then mapped to the bounding box defined for the geometric container, which is the cone.

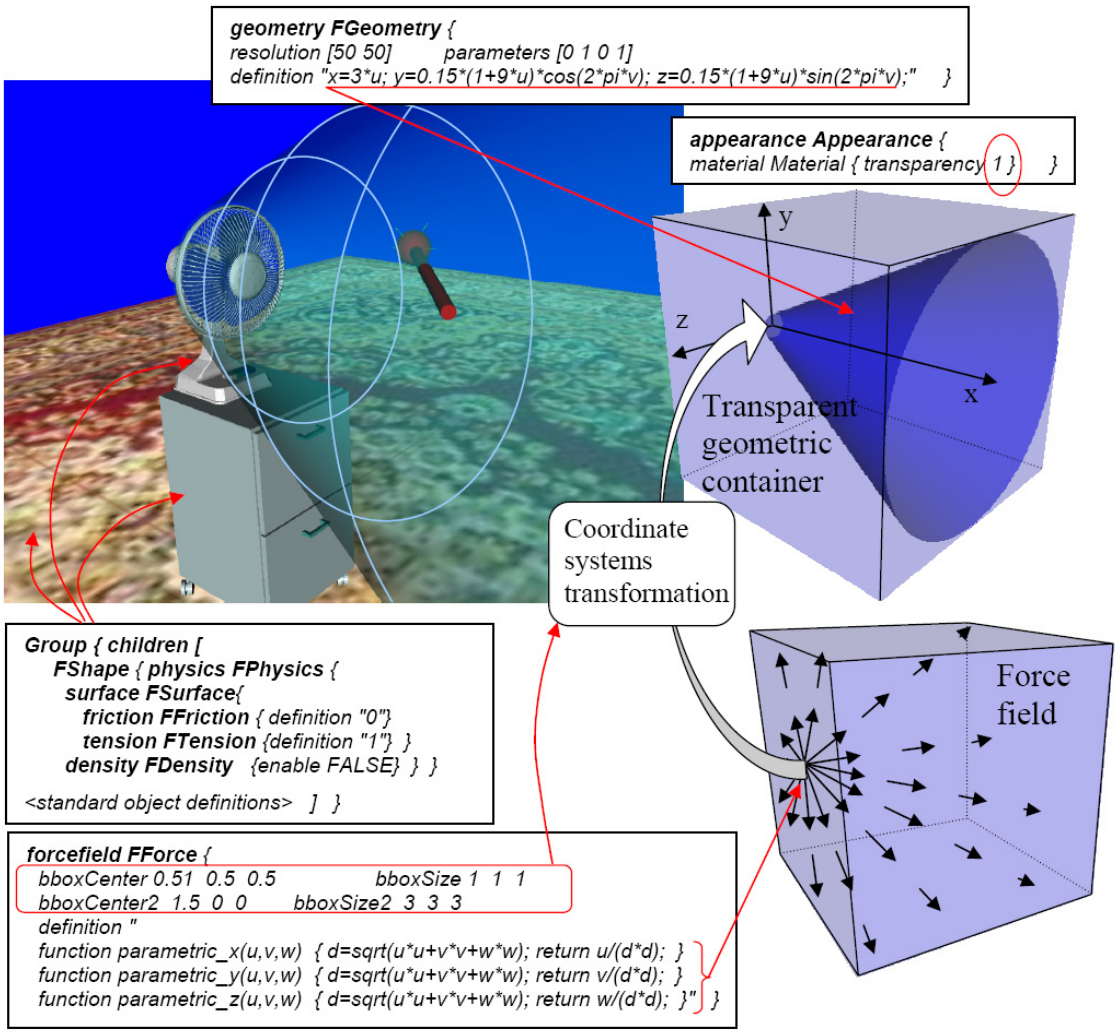


Figure 7.2 Modeling “Tangible Wind” using invisible force field

In the next example, we make an invisible implicit cylinder within which a helical centripetal force field is defined. The force filed defines a whirlpool moving counter clockwise and down along the axis of the cylinder (Figure 7.3). A standard X3D scene of the pool with a moving texture image is called from the online Web3D repository to superimpose with the force field. In the resulted scene, the haptic device will feel the flow of the water inside the pool as a force actively rotating the handle counter

clockwise around the vertical axis and down the pool. The simplified code of the example is illustrated in Figure 7.4.

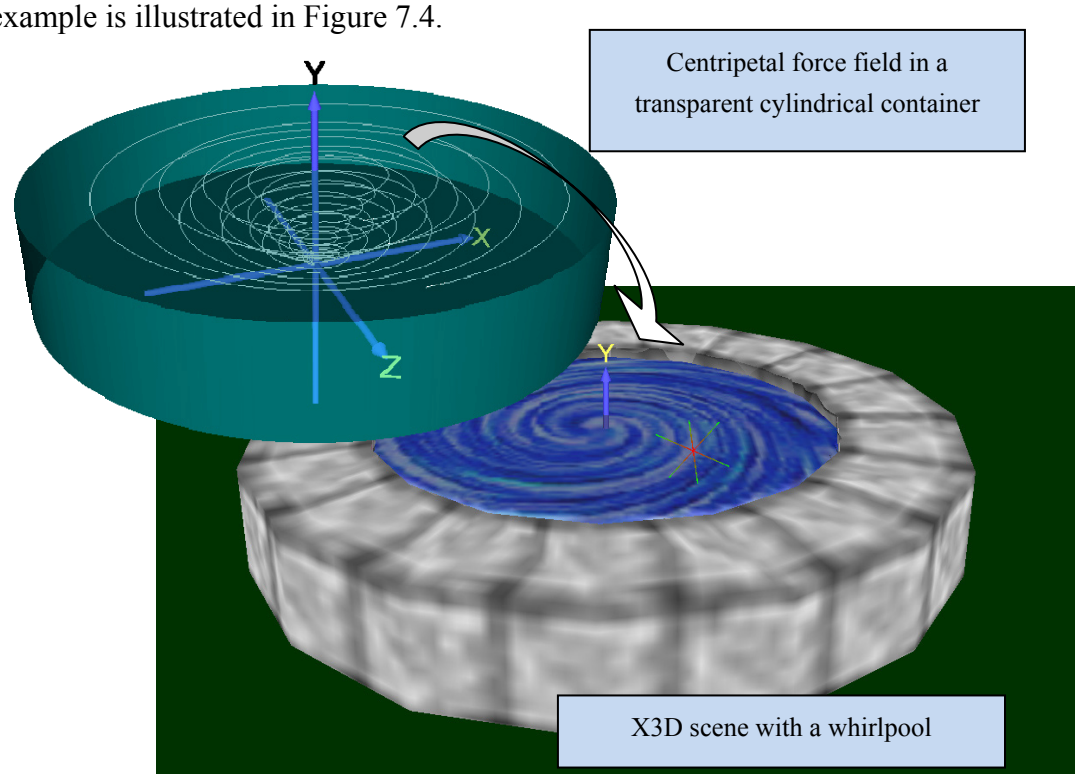


Figure 7.3 A cylinder with a helical force field inside it

```
FShape {
   geometry FGeometry {                    Invisible cylindrical container
       definition "(1.6^2-x^2-z^2) &y& (0.7-y)"
       bboxCenter 0 0.35 0        bboxSize 3.2 0.7 3.2
} appearance FAppearance {material FMaterial {transparency "0"}

}
   physics FPhysics {
       friction FFriction {enable FALSE}
       force FForce {        definition"          Definition of a force vector
              function parametric_x (u, v, w)
{d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return x/sqrt(x^2+y^2+z^2) ;}
              function parametric_y (u, v, w)
{d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return y/sqrt(x^2+y^2+z^2) ;}
              function parametric_z (u, v, w)
{d=-0.1; x=w+d*u; y=d*v; z=-u+d*w; return z/sqrt(x^2+y^2+z^2) ;}"
          parameters [-1 1    0 1    -1 1]                    }
}                    }
                                  Link to an X3D scene with a moving texture whirlpool
Inline {url "http://.....MovieTextureWhirlpool.wrl"}
```

Figure 7.4 Simplified code of Figure 7.3

Another example of defining tangible objects in an X3D scene is given in Figure 7.5. Here, an X3D model of a moving shark, which is linked (inlined) from the companion web resource for the X3D reference book [139], is turned into a tangible solid object by associating (grouping) it with the function-defined physical properties which are set as a medium-tension surface with friction and non-uniform density inside it (softer at the peripheral parts and harder when closer to the shark's backbone). Besides this, a time-controlled force field is defined around the shark's tale. The force vector follows the motion of the tale which is defined within a cycle interval of 5 sec. To feel this force, the haptic actuator has to be placed inside an invisible geometric container which is a box defined by FRep functions as an intersection of 6 half-spaces bounded by planes. The box's appearance is defined as invisible to the viewer (transparency 1). The force vectors are defined by parametric functions for each point inside the box defined by their Cartesian coordinates (u, v, w) and time t. Infinitely repeating 5 sec time intervals (cycleInterval 5) in the virtual scene map to time intervals [0, 1] within the parametric definitions of the force vector (timeSpan 0 1). Since the time in the scene is controlled by the computer timer and the same time interval is used both for the force and in the model definition of the shark, the motion of the tail will be synchronized with the forces simulating water pressure created on the sides of the tale when it waves. The magnitude of the force vector increases for the points closer to the end of the tale and decreases with increasing the distance from it. For illustration purposes, the box and the force vectors are shown in the figure as a transparent blue box and black arrows, respectively. The simplified code of the example is illustrated in Figure 7.6.

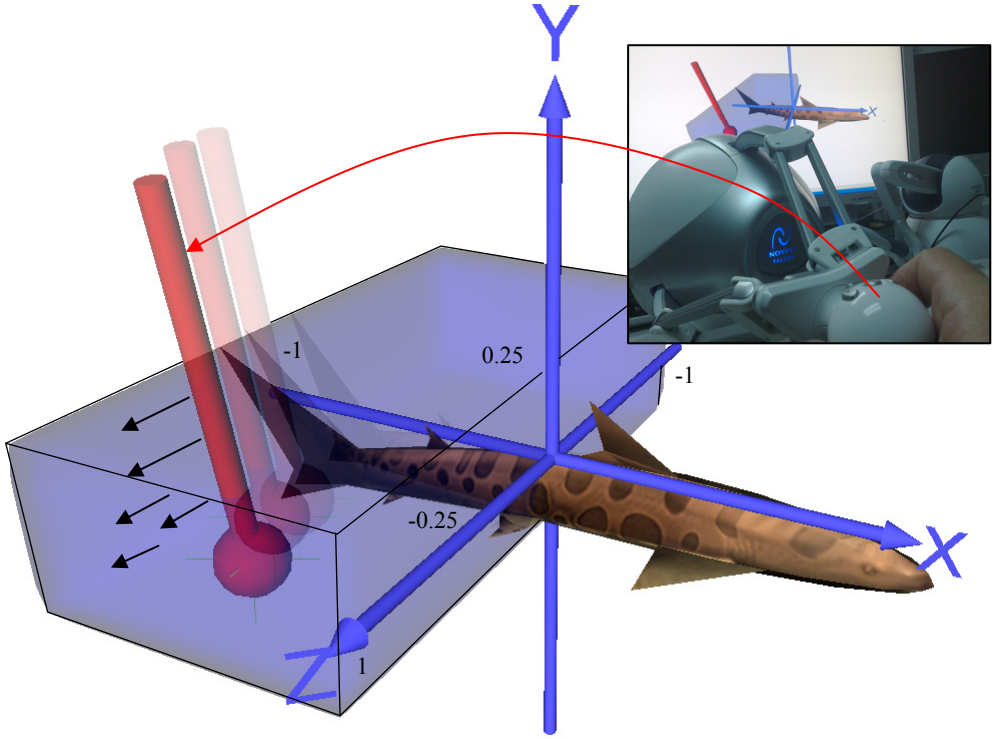


Figure 7.5 Modeling “Moving Shark” using density and time-dependent force field

```
<! header and prototypes are skipped >
Group {children [Inline {url "http://x3dgraphics.com/examples/X3dForWebAuthors/
KelpForestExhibit/SharkLucy.x3d"}

FShape {
        physics FPhysics {surface FSurface {tension FTension {definition "0.8”}}
                          density FDensity {definition "function frep(x, y, z)
                  {dist=sqrt  ((y+0.125)  ^2+z^2);  d=0.2;  if  (dist<0.05)  d=1;  return
         d ;”}}}]}

FShape {
        geometry FGeometry {definition "(x+1)&(0.25-y)&(y+0.25)&(z+1)&(1-z)&(-x)" }
        appearance Appearance {material Material {transparency 1}}
        physics FPhysics {forcefield FForce {cycleInterval 5    timeSpan 0 1

     definition”
           function parametric_x (u, v, w, t) {return 0 ;}
           function parametric_y (u, v, w, t) {return 0 ;}
           function parametric_z(u, v, w, t)
                          {return –u*(1-2*abs (2*t-1))/ (1+10*abs (w)) ;}"
     parameters [-1 1 -1 1 -1 1]     }}
}
```

Standard X3D object is inlined and associated with physical properties

Invisible box for the force field

Parametrically defined force vectors for any given 3D coordinates *u*,*v*,*w*, and time *t* cycled by 5 sec

Figure 7.6 Simplified code of Figure 7.5

7.2 Practical collaborative applications

7.2.1 Thin server-based collaborative applications

Compared with typical collaborative applications which concentrate on visual interactions, our thin server-based collaborative applications could also support haptic interactions in shared virtual spaces. Therefore, collaborative users are able not only to see but also to touch the virtual objects, as well as to collaboratively modify their visual and haptic properties in shared scenes. To set up a thin server-based collaborative application, the developers have only to use a template of the core VRML or X3D file which establishes the link with the server, defines the login and chat modules and finally links to the VRML or X3D codes of the shared space.

In Figure 7.7 the thin server architecture is used to set up Virtual Campus of Nanyang Technological University. Account management is implemented using PHP scripting language and MySQL database which supplies the user's parameters to the root file upon successful login, including the parameters of avatars, voice features and text/text-to-speech chat option. The scene files of the Virtual Campus are originally standard VRML/X3D files. By integrating several necessary function modules, one root scene file and a corresponding physical property node, all scenes become shared and tangible. In Figure 7.7 the collaborative users could examine and interact with various places such as the main building, tutorial room and student hall both visually and haptically. Depending on the size and locations of the scenes, the haptic zone can be adjusted accordingly to best fit the user's touching experience. A bigger haptic zone allows the user to have a larger interaction area (e.g., to examine the student hall room)

while a smaller haptic zone could provide for a higher precision for haptic interaction (e.g., to feel the moving curtain). A simplified code of the Virtual Campus is illustrated in Figure 7.8.

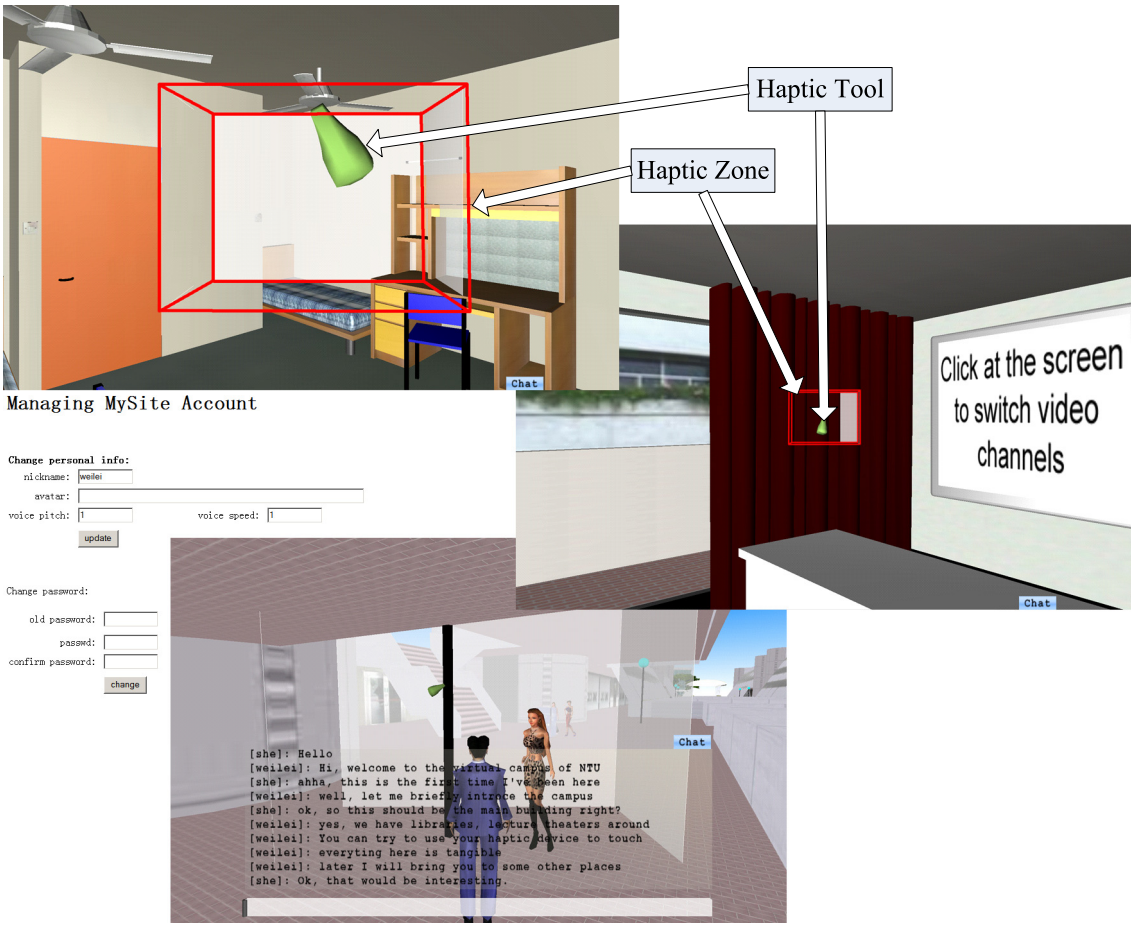


Figure 7.7 Setting up “Virtual Campus” based on thin server architecture

```
DEF Conn NetConnection
{
      address     "cospace.sce.ntu.edu.sg"
      port        12345
      protocol    3
}
DEF MU BSCollaborate
{simplified }

Group
{
   FShape
   {
        geometry FGeometry {definition: "0"}
        physics FPhysics {
             surface FSurface {
                  tension FTension {enable TRUE definition: "1"}
                  friction FFriction {enable TRUE definition: "0.3"}
             }
        }
   }
   Inline {url    "http://cospace.sce.ntu.edu.sg/vircampus/enter.wrl"}
}
//----------------------Haptic routine Connector-------------
PROTO HapticDevice
[simplified]
DEF HapticInstantiation Script
{simplified}
//----------------------Chat Modules--------------------------
EXTERNPROTO Chat
[simplified]
"Chat.wrl#Chat"
DEF ChatScript Script
{simplified}
//----------------------TTS Modules---------------------------
DEF TTS TextToSpeech
{simplified }
DEF ScrTtsMgr Script
{simplified }
//----------------------Avatar Modules-------------------------
PROTO AvatarInfo
[simplified]
DEF ScrAvatarMgr Script
{simplified}
```

Necessary information for connection to the server

Add physical properties to all geometries in the Virtual Campus to make them tangible

Scene root file which links to all other scene files

Initialize connection with local haptic device if possible

Optional modules for enhanced functionalities

Figure 7.8 Simplified code of Figure 7.7

In Figure 7.9, a screenshot of the educational shared interactive tool for modeling geometric shapes defined by mathematical formulas is displayed. The geometry, appearance and physical properties of the objects can be defined in a uniform way and then interactively rendered and modified. The shared model can be loaded from a file stored on a web server or manually created by user commands. In Figure 7.9 the geometry loading procedure is done by the user *"Student"* who enters an implicit definition of a mechanical assembly. The model colour is by default neutral grey and therefore user *"WeiLei"* applies new colour definition onto the geometry. Since the shared model is created without physical properties, user *"Student"* then issues a command to set density to the shared geometry. After all visual and physical properties have been defined properly, each user will need to create a customized tool avatar to be able to interact with the shared model. This can be done either by typing mathematical formulas in the right panel, or by choosing the tool avatar they want from the pre-defined tool list. Colours of the tool avatars can also be set in the right panel. Since this collaborative application is based on the thin server architecture, only the modeling result will be synchronously updated to all users, the user's tool avatar will only be seen by its owner. Therefore user *"WeiLei"* chooses a big cylinder as his tool avatar and overlaps his tool avatar on the shared geometry to modify it. User *"Student"* uses his small cylinder tool avatar to dig several holes on the shared geometry. Similarly, they then collaboratively change the friction and tension of their respective tool avatars and use them to further modify various properties of the shared model. After the modeling procedure has been done, they are able to examine all modifications either done by themselves or by other collaborative users. Considering the fact that not every user has a haptic device, either haptic device or a

mouse could be used to position the tool avatar and apply new visual and physical properties onto the shared model, however only those users with haptic devices could feel and interact with the physical property modifications. The simplified interaction procedure is illustrated in the command window of Figure 7.9 as well as in Figure 7.10

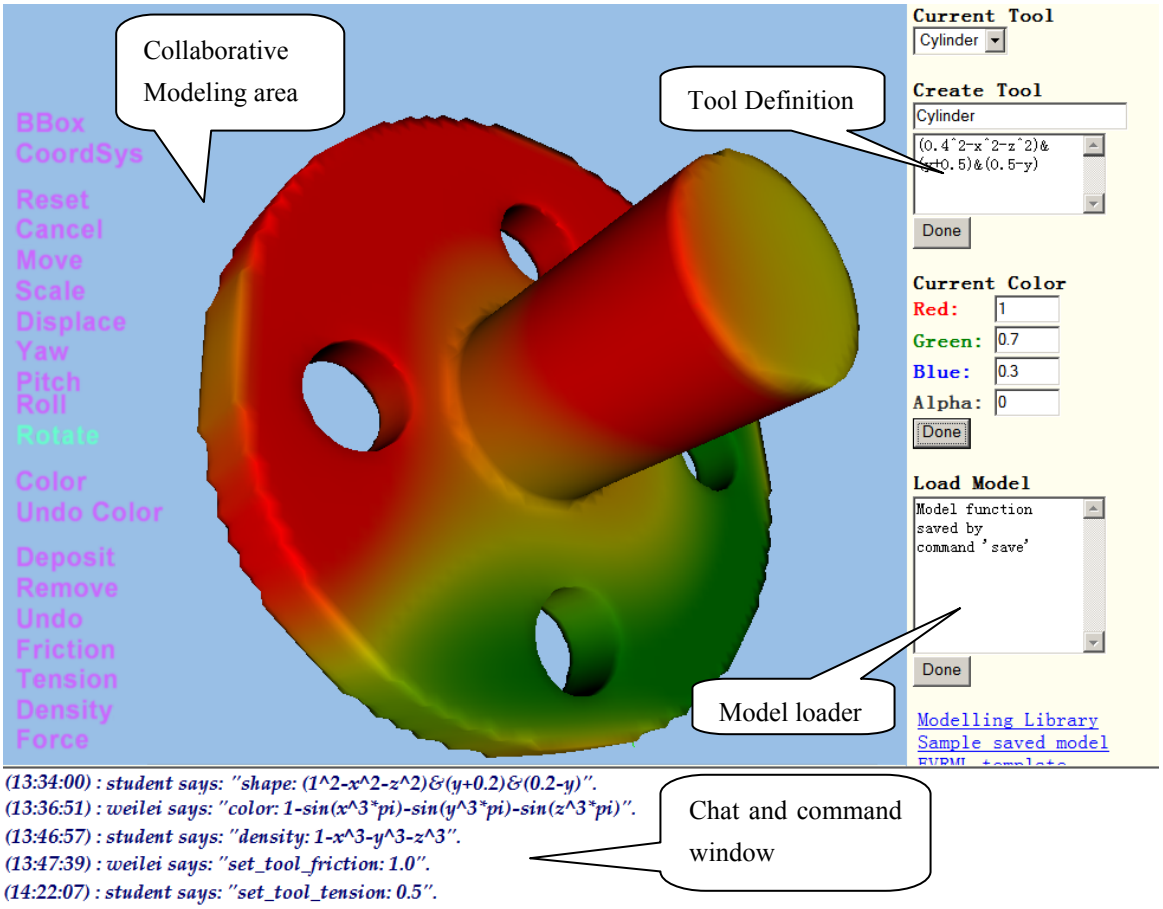


Figure 7.9 Thin server based collaborative applications

Student: shape: {simplified} //load shared geometry, the default colour is neutral gray
WeiLei: colour: {simplified} //add colour onto the shared geometry
Student: density: {simplified} //add physical properties onto the shared geometry

WeiLei: //create big cylinder as tool avatar and change friction value
Student: //create small cylinder as tool avatar and change tension value
WeiLei: //apply friction modification on the shared geometry using big cylinder
Student: //apply tension modification on the shared geometry using small cylinder

Student: save //save the model

Figure 7.10 Simplified interaction procedure in Figure 7.9

A more advanced application example of using the developed collaborative framework on interactive segmentation of brain MRI data is illustrated in Figure 7.11. It is developed using the FRML/FX3D and native script dll functions. Here, the model of a human brain is reconstructed from the MRI data and collaboratively explored and edited by several networked users. To check the correctness of the calculated brain surface with reference to the actual MRI volume data, a spherical sampling tool is used. The colours on the sphere are calculated with a function which maps the actual MRI density to the colour. Avatars of the tools of the concurrent users are displayed as 3D crosshair markers. The polygonal brain surface with the colours marking probable segmentation errors is calculated in the binary functions stored in DLLs and then sent back to VRML rendering pipeline for visualization. Interactive editing of the sphere is done in a pair of VRML script and the respective DLL function where the computations are performed in the DLL and the interaction in 3D at the VRML script level. The DLL functions communicate through the common shared memory. The surface of the brain is declared tangible to feel it with the optional haptic device. Collaborative users see each other's sampling tool as a crosshair while their own sampling tool is displayed as a sphere with dynamically changing colours. The users are also able to see how the surface of the brain is changing while being edited by all the participating parties. In comparison with other collaborative shape modeling framework, their haptic atomic operations are based on objects. There is no easy way for them to apply a haptic modification which affects only certain parts of an object. This feature is an obvious advantage of the integrated function-based approach.

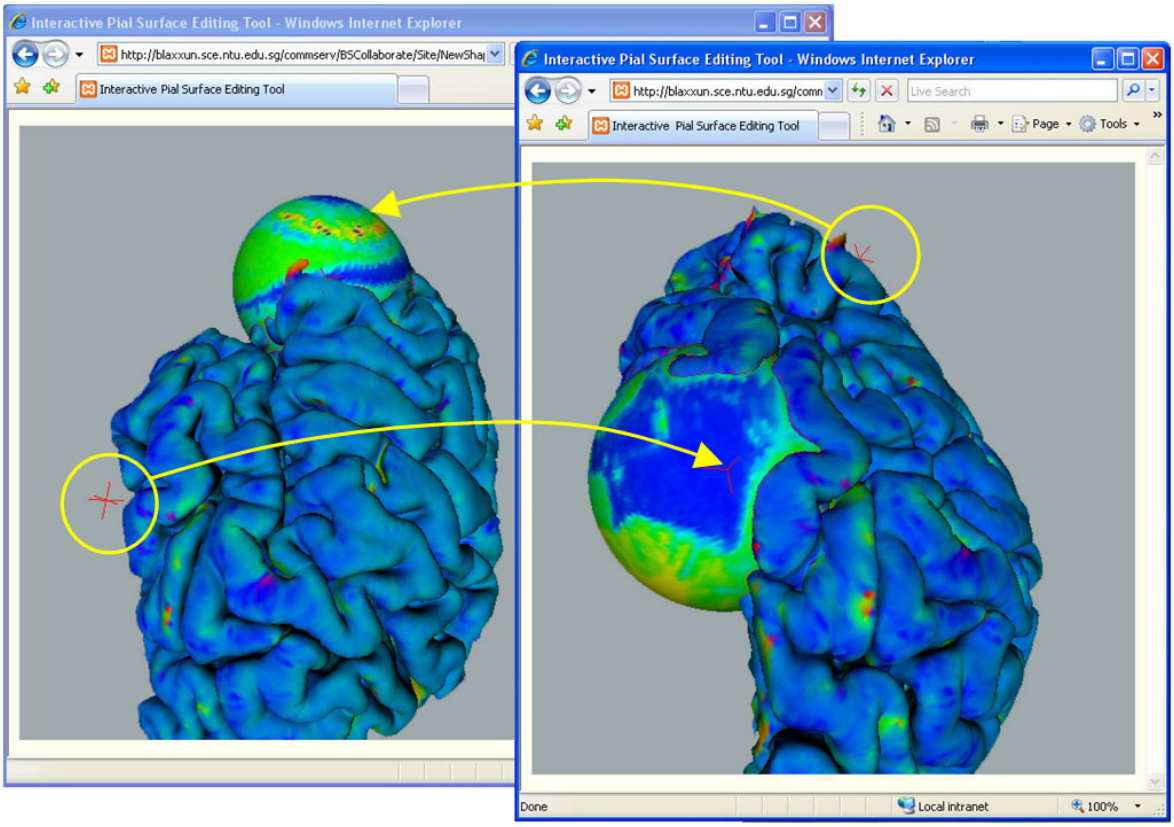


Figure 7.11 An advanced example of a collaborative scene with the thin server used

7.2.2 Strong server-based collaborative applications

Thin server-based architecture could provide functionalities for haptic interaction in shared virtual spaces, while the real haptic collaboration, where the users can actually feel each other, can only be done based on the strong server-based architecture. By using strong server, each user's tool avatar is also considered as a shared object and therefore they can be seen and touched by each other. To develop practical collaborative applications with the strong server, the users have to use VRML/X3D code template with several prototypes, which establish the link with the server and its database, and also define the URLs and a few other parameters of the VRML/X3D objects which will be used as tools in the scene. The shared tool prototype will efficiently synchronize complex interactions. It has independent *EventStreamSensors* for each collaborative field, therefore

the tool avatars can respond to very complex interactions in real time including the haptic interaction synchronization, which updates at 1000 Hz. The shared object libraries prototype is needed to efficiently create and synchronize complex shared object interactions. For example, generation and storage of shared object instances and shared object manipulations (attach, detach, exchange, and compete). The internal database on the server keeps all previous collaboration state even after all users have left.

In Figure 7.12, we show two networked users working with the same collaborative scene with several shared objects defined in it. The left user is moving the fancy function-defined object while the right user is using a standard VRML/X3D model of a house as a tool. The objects displayed in the figure are turned into tangible by associating them with the physical properties. Hence, they can be felt by the networked haptic devices. If any of the shared objects is assigned to be a tool by linking from the shared object modules of the root file, it will then move following the motion of the respective device and all the users will synchronously see its motion. Each user can have one or several tools concurrently which have to be associated with the respective interactive or haptic devices. If the tool shape has tangible physical properties, other users will be able to feel it as well as other tangible objects.

When a shared object is selected as a tool, either strong or weak locking can be applied either prohibiting or allowing other users to reclaim this object as a tool. Hence, in Figure 7.13 two clients with haptic devices share the same tool object. It results in synchronized motion of the networked haptic devices and coordinated motion of the tool on the client screens. The users are able to physically feel each other's motion. A simplified code for Figure 7.12 and Figure 7.13 is illustrated in Figure 7.14.

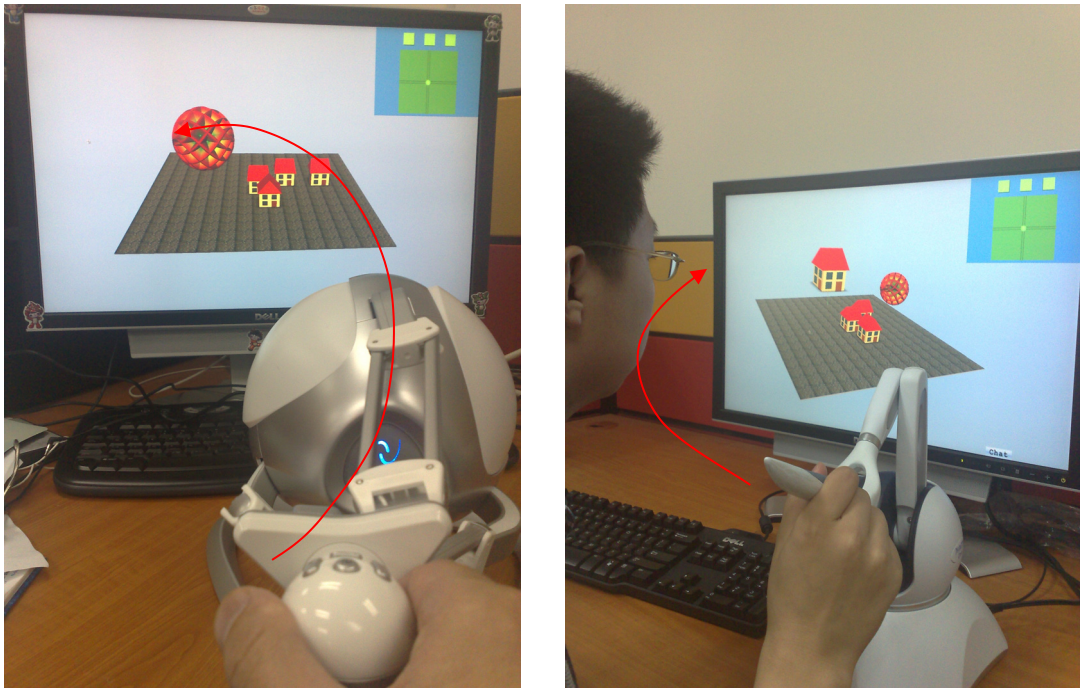


Figure 7.12 Collaboration of two users in a strong server based scene

Figure 7.13 Synchronized motion of the haptic devices associated with the same tool object

```
DEF Conn NetConnection
  {
      address     "cospace.sce.ntu.edu.sg"
      port        12345
      protocol    3
  }
  DEF MU BSCollaborate
  {simplified }
  Inline {url  "http://cospace.sce.ntu.edu.sg/strong_server/basic_examples.wrl"}
  //----------------------Haptic routine Connector-------------
  PROTO HapticDevice
  [simplified]
  DEF HapticInstantiation Script
  {simplified}
  //----------------------Chat Modules--------------------------
  EXTERNPROTO Chat
  [simplified]
  "Chat.wrl#Chat"
  DEF ChatScript Script
  {simplified}
  //----------------------TTS Modules---------------------------
  DEF TTS TextToSpeech
  {simplified }
  DEF ScrTtsMgr Script
  {simplified }
  //----------------------Avatar Modules---------------------------
  PROTO AvatarInfo
  [simplified]
  DEF ScrAvatarMgr Script
  {simplified}
  //---------------------Default Tool Avatar Modules---------
  PROTO HapticCursor
  [simplified]
  DEF CursorMgr Script
  {simplified}
  //----------------------Shared Object Modules--------------
  PROTO SharedObj
  [simplified]
  DEF SharedObjMgr Script
  {simplified}
  DEF SharedObjAppLogic Script
  {simplified}
```

Necessary information for connection to the server

Scene root file which links to all other scene files

Initialize connection with local haptic device if available

Optional modules for enhanced functionalities

Default tool avatar definition used for application initialization

Shared object pool, type declaration, instance creation and destruction, shared object re-routing, lock mode selection and other application logic

Figure 7.14 Simplified code of Figure 7.12 and Figure 7.13

Figure 7.15-Figure 7.19 show a collaborative haptic shape modeling session. The shared object to be collaboratively modified is displayed as a human femur defined in VRML/X3D. To provide faster collaborative haptic interaction, a PDE method for patchwise approximation [127] is adopted to reduce the number of polygons on the polygonal model. The physical properties of the femur model are defined by combining a group of invisible standard VRML and FVRML shapes with it, as shown in Figure 7.15.

In the graphical user interface of this application, there are mainly three parts: the shared 3D modeling scene, the control panel and the command/chat window. The 3D haptic modeling scene and the chat area are shared amongst 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 messages in the command/chat panel concurrently. To change the current tool avatar, the pre-defined tools and existing customized tools can be chosen from the drop-down menu in the control panel. Different tools generate different visual and physical modification results, for adding new shared objects into the collaborative scene, the users could click on the green buttons in the upper right corner to generate an instance of one type of the shared objects.

Two collaborative users are logged in to the modeling session, user A works with Sensable Phantom Omni device while user B uses Novint falcon device. Both of them use a human hand model as the default tool avatar (Figure 7.16). The users are supposed to collaboratively insert a screw into the femur through a plate. Corresponding haptic property modifications should be done as well.

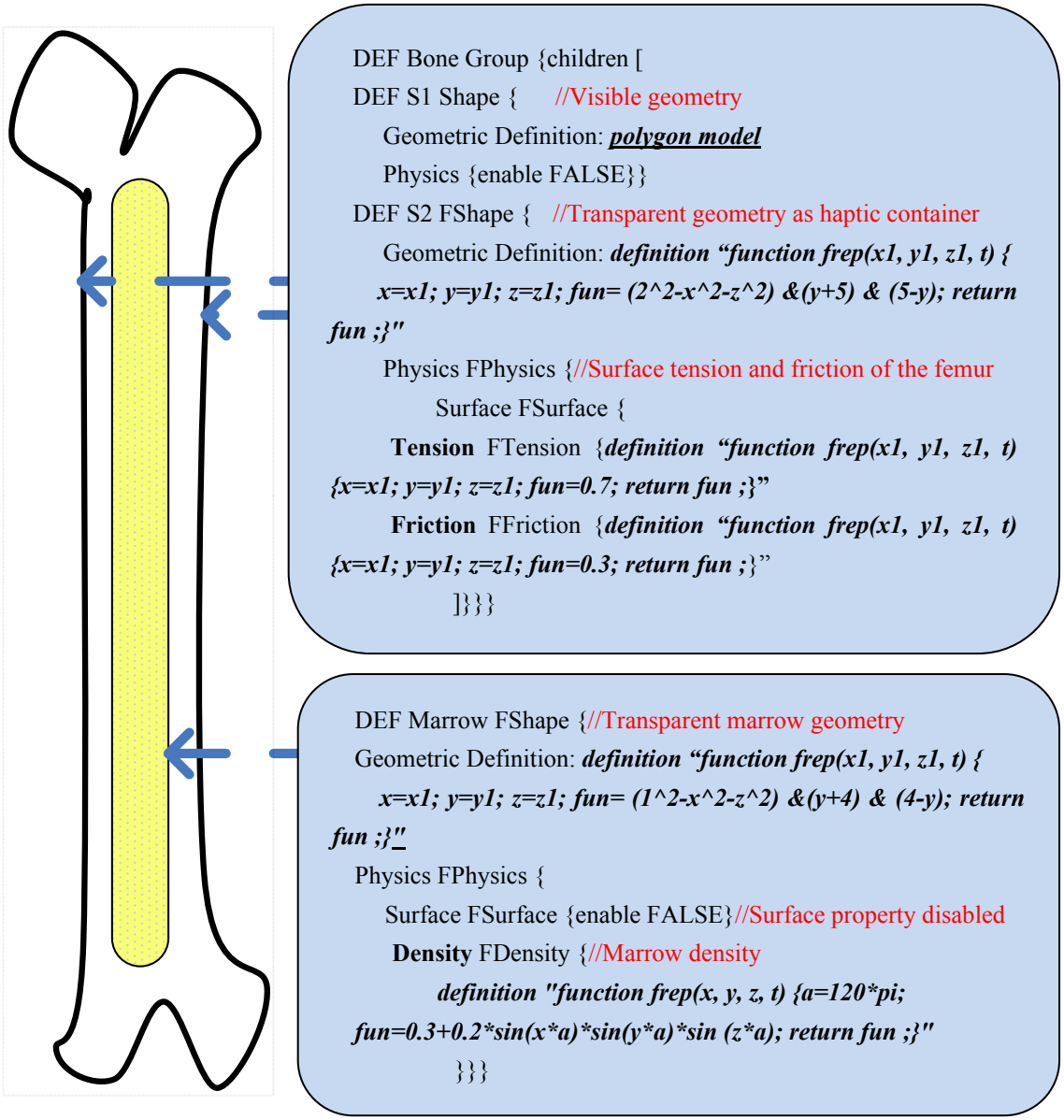


Figure 7.15 haptic properties defined by superimposing a group of invisible objects onto the femur

The collaboration procedures are illustrated as follows:

1. Both users will first acquire their new tool avatar for the modeling session. Therefore they choose from the right panel their new tool avatars and apply them. After that, they adopt a screw and a plate respectively for further modeling.
2. In order to insert the screw into the femur, both users will have to set physical properties on their respective tool avatars. Therefore user A will first type "set_init_tool_tension 0" in his command window to simulate the emptiness of the

hole on the femur surface after drilling, To simulate the hardness of the metal screw and plate, both users will type “set_init_tool_density 1” in their respective command windows (Figure 7.17).

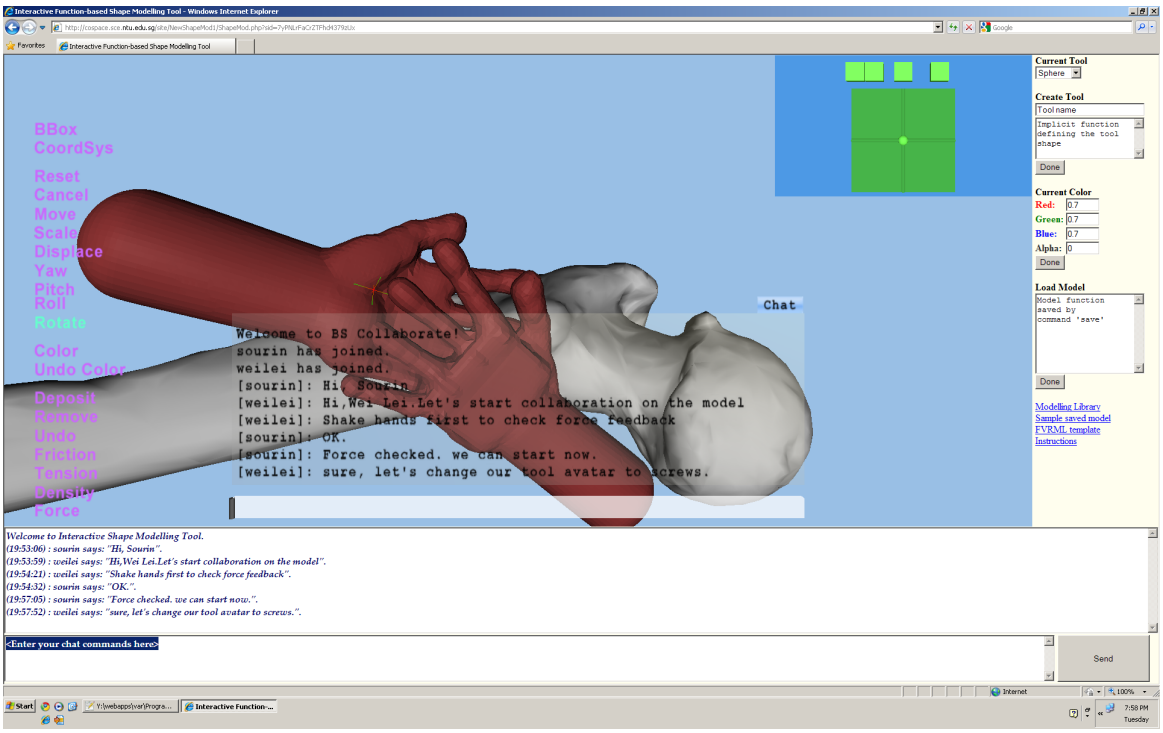


Figure 7.16 Collaborative shape modeling GUI illustration

User B will have to position the plate at the right place on the femur surface first. User A will then use his screw avatar to insert through one hole on the plate. Since both users could feel each other’s tool avatar as well as the tension of the femur, this collaborative positioning procedure could be done easily. The actual drilling operation is based on set operations between function-based haptic containers of the femur and the screw; however such procedure is totally transparent to users. Although the haptic containers exceed the actual polygonal surface (i.e. the cylinder for surface tension and friction is larger than the femur geometry), the physical properties rendering will be activated only within the geometry.

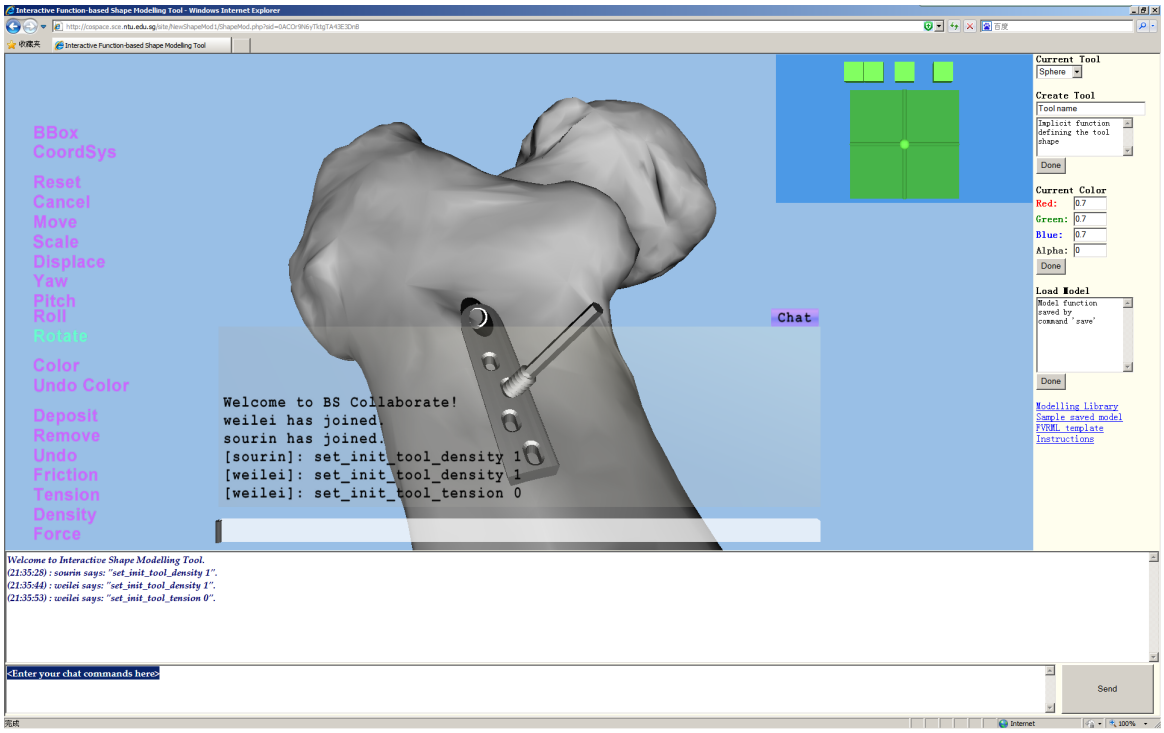


Figure 7.17 Select tool avatar and redefine tool properties

3. Once User A has applied the drilling operation, the drilling point on the femur surface will have no tension and is totally penetrable, and the drilled hole will have maximum density inside to simulate the inserted screw, just as the screw's haptic property was defined previously. New instances of the screw and plate will be generated and stay with the femur.
4. User A and user B examine their collaborative modeling work. User A is not satisfied with the insertion, and therefore he released his screw avatar and grabbed the plate from user B. This is applicable because weak lock is enabled. The previous drilling operation is cancelled, and this time user A will position the plate while user B will drill.
5. Another drilling operation is done at a different place on the femur surface. The two users check the result again with both vision and touch. This time both of them are

satisfied with the modeling result. The modified femur model is then exported and saved, along with all the visual and haptic properties (Figure 7.18). The simplified code of the exported model definition is illustrated in Figure 7.19.

By illustrating the collaborative shape modeling procedure step by step, we could clearly see that such strong server-based collaborative application could provide great flexibility through the whole modeling procedure, and the result produced could be as complicated as the users could propose with mathematical functions and procedures.

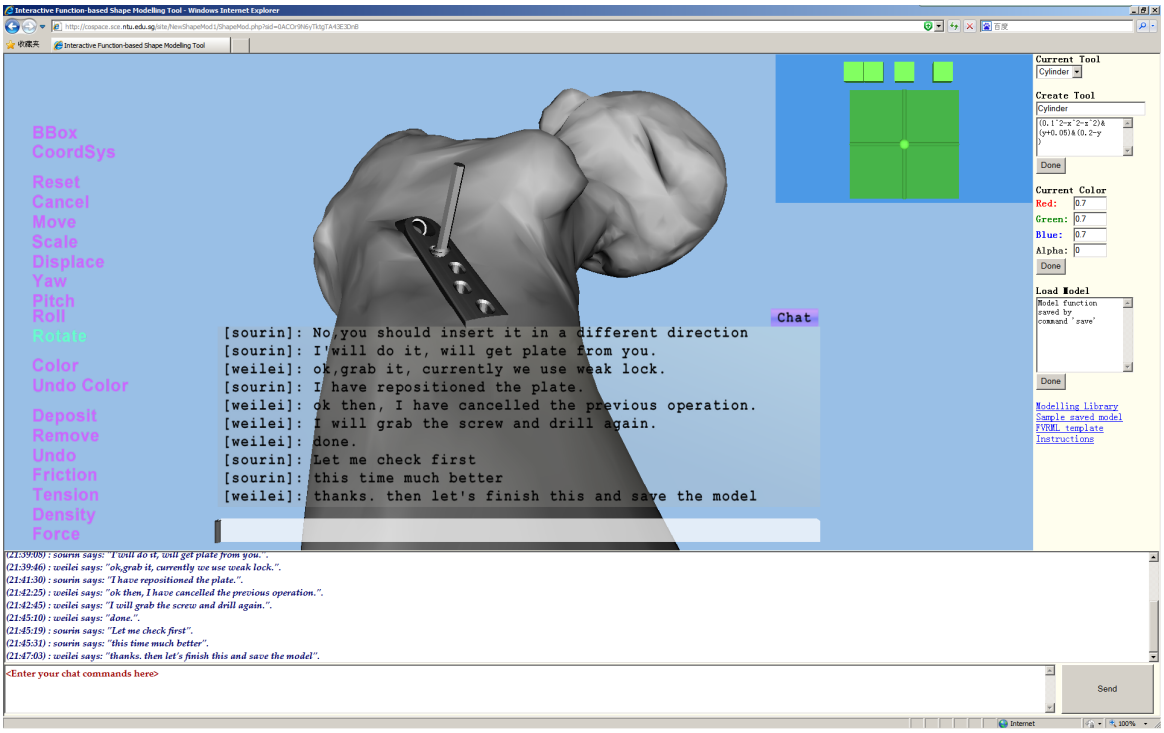


Figure 7.18 Cancel previous operation, grab shared object and reapply

```
DEF Bone Group {
   children [

        DEF S1 Shape {
            Geometric Definition: femur model+plate model+screw model
            Physics {enable FALSE}
            }

        DEF S2 FShape {
            Geometric Definition: definition "function frep(x1, y1, z1, t) {
             x=x1; y=y1; z=z1; fun= (2^2-x^2-z^2) &(y+5) & (5-y); return fun ;}"
            Physics FPhysics {
                Surface FSurface {
                    Tension FTension {
                        definition "function frep(x1, y1, z1, t) {
                        x=x1;y=y1;z=z1;fun=0.7;
   x=x1*(-0.5809)+y1*(-0.5459000000000001)+z1*(-0.6038)+(2.1899);y=x1*(0.733)+
y1*(-0.0282)+z1*(-0.6797)+(0.6789);z=x1*(0.354)+y1*(-0.8374)+z1*(0.4165)+(3.7112);
if((2^2-x^2-z^2)&(y+5)&(5-y)>=0){ fun = 0;}return fun;}"
                    }
                    Friction FFriction {
                        definition" function frep(x1, y1, z1, t) {
                    x=x1; y=y1; z=z1; fun=0.3; return fun ;}"
                    }
        ]
   }
   }
   }

   DEF Marrow FShape {
      Geometric Definition: definition "function frep(x1, y1, z1, t) {
       x=x1; y=y1; z=z1; fun= (1^2-x^2-z^2) &(y+4) & (4-y); return fun ;}"
      Physics FPhysics {
          Surface FSurface {enable FALSE}
          Density FDensity {
              definition "function frep(x, y, z, t) {a=120*pi;
              fun=0.3+0.2*sin(x*a)*sin(y*a)*sin(z*a);
   x=x1*(-0.5809)+y1*(-0.5459000000000001)+z1*(-0.6038)+(2.1899);y=x1*(0.733)+
y1*(-0.0282)+z1*(-0.6797)+(0.6789);z=x1*(0.354)+y1*(-0.8374)+z1*(0.4165)+(3.7112);
if((1^2-x^2-z^2)&(y+4)&(4-y)>=0){ fun = 1;}return fun; }"
              }
   }
   }
```

Transparent geometry

New Femur tension definition after drilling operation

Transparent geometry

New Marrow density definition after drilling operation

Figure 7.19 Simplified code exported after the collaborative shape modeling session

7.3 Summary

In this chapter, several comprehensive examples and practical collaborative applications are presented to illustrate the features provided by the unified modeling paradigm and collaborative framework. Both of them are proven to be useful for real life applications. The unified modeling paradigm provides a way to define various object properties within one model and render them through different output devices. The modeling framework enables geographically separated users to collaboratively work within one shared environments and interact with each other as well as with shared objects and tools. Compared with other approaches, the most obvious advantages here are the compactness, freedom and variety through the whole shape modeling procedure with the incorporation of mathematical functions and procedures. However, we have to admit that such paradigm and framework are not applicable to all situations. For example, they are not very efficient in working with pure polygonal representations.

Some of the examples illustrating haptic interaction and collaboration can be seen at http://www3.ntu.edu.sg/home5/weil0004/Video.html (Figure 7.20).

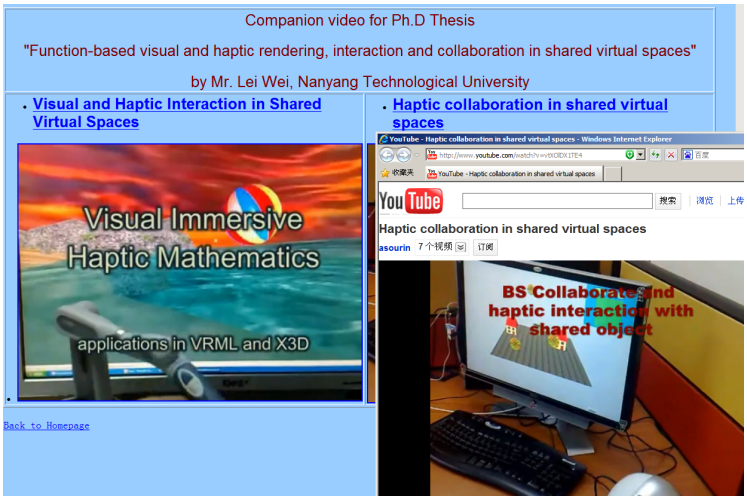


Figure 7.20 Companion video for illustrating haptic interaction and collaboration

Chapter 8 Conclusion and future work

This chapter summarizes the work presented in this thesis. In Section 8.1 the conclusion is made, and future work is discussed in Section 8.2.

8.1 Conclusion

After finishing a comprehensive survey of the relevant literature as well as a rigorous study of the current research and industrial projects in the area of shared virtual reality and haptic interaction, the research niche for the development of the function-based approach to collaborative haptic shape modeling and interaction in shared virtual spaces had been identified and motivated the subsequent R&D work.

In Chapter 1 we hypothesized that using mathematical functions in virtual object definitions would greatly help to collaboratively model and render them as well as to haptically interact and collaborate in the shared virtual scenes. We will prove that this PhD project has defended the above hypothesis by the following three properties:

Behaviour: In this PhD thesis, we have proposed and implemented a visual and haptic collaborative framework which supports using mathematical functions and procedures to define virtual object properties in shared virtual scenes. Various object properties such as geometry, appearance and physical properties could be interactively defined and modified using implicit, explicit and parametric analytical functions. These function-based definitions could then be accurately and quickly transmitted within the collaborative environment. The incorporation of haptic interaction is considered from the very beginning of the project. Therefore both visual and haptic interactions are naturally

compatible with the collaborative framework and greatly enhance the immersion of the system. In all, our solution to the objective is quite efficient, accurate and reliable.

Coverage: The proposed solution could be applied to a wide range of applications such as edutainment, medical training simulation and data visualization. The system is also compatible with different application configurations ranging from single-computer haptic interaction to haptic collaboration.

Efficiency: The resource consumed by the proposed system is moderate. Any mid-level desktop computers produced within the past 4 years are sufficient for the system. Our implementation requires less than 3MB hard disk space and less than 15 seconds to install. The client program requires about 150MB memory when running single-computer interaction and 200 MB when running collaborative interaction. Such requirements are quite affordable with modern desktop computers which are often equipped with 4GB – 8GB memory. Our system has no specific requirement for video cards.

The PhD project presented in this thesis has resulted in the following main research and development novelties:

1. **A novel modeling paradigm** for definition of the virtual objects' geometry, its visual appearance and tangible physical properties has been proposed and detailed. To efficiently exchange models in cyberspace, the properties of the objects can be defined by mathematical functions and algorithmic procedures which are both definitions and implementations of the properties. This approach is independent of the rendering and interactive system which can be used for its implementation.

2. **A novel haptic collision detection and rendering algorithm** has been devised as a core part for the implementation of the proposed modeling paradigm. The algorithm permits to define tangible physical properties for the wide class of geometric objects with and without a surface defined. The algorithm also permits to haptically explore virtual scenes with mixed model content and any initial location of the haptic tool. Both surface and inner physical properties can be freely used for various virtual objects – defined by mathematical functions as well as many common models. The algorithm caters for various relevant sizes of the objects used in the scenes as well as provides for a user-controlled precision of the haptic interaction.
3. **An innovative visual and haptic collaborative framework for shared virtual spaces** has been designed and implemented. The framework is based on server-client architecture and comes in two variants: thin and strong server. These frameworks are designed for light-weight and advance haptic collaborative applications, respectively. As a part of this work, we have enriched the concept of shared virtual objects as well as proposed and further studied the concept of virtual haptic tools and relevant topics of shared events, synchronization and consistency control. The proposed framework is designed to work very efficiently with function-defined models due to their small size, however it can be applied to any other models as well, as it has been further experimentally proved in the project.
4. **A few representative implementations and comprehensive programming examples** have been programmed using the international open standard software VRML and X3D and presented in the thesis as a proof of the proposed concepts and to verify the correctness of the devised algorithms and developed core software parts.

The research and development results achieved have been published in **5 journal and 6 conference papers**, which are listed in Appendix "List of Literature". The research results as well as the software developed heavily contributed to the large **on-going externally funded project** "Visual and Haptic Rendering in Co-space". The developed software was showcased at the exhibition Discover Engineering @ NTU 2009. The software is also being used for teaching NTU CSC204/CPE411 "Computer Graphics" students as a part of the coursework "Implicit Fantasies and Parametric Metamorphoses in Cyberworlds".

Some of the examples illustrating haptic interaction and collaboration can be seen on the following web page: http://www3.ntu.edu.sg/home5/weil0004/Video.html .

8.2 Future work

While working on the proposed modeling paradigm and collaborative framework, several areas of future research have been identified.

8.2.1 Physics engine-based visual and haptic interaction

Physics engines are originally introduced for realistic visual rendering. They could be software, hardware or a combination of both. Physics engines are focused on simulating both physics models and other pseudo-physical models by using various physical terms such as mass, velocity, friction and resistance, which opens prospects of cooperation between physics engines and haptic devices.

However, such cooperation still has certain difficulties. First, the physical calculation models in physics engines are usually predefined; therefore it would be quite troublesome

if specific rendering simulation is needed which requires the physical calculation models to adapt to extra parameters. Second, Physics engines are designed to work mainly for visual rendering, therefore the rendering procedure is a closed loop that only requires fake force input from mouse and keyboard, and no real force will be calculated and sent out beyond the visual pipeline. For applications that require interactions across visual and haptic pipeline, physics engines could not be directly used.

Nevertheless, the trend of using physics engines for haptic interaction is growing fast. Considering the fact that certain VRML/X3D browsers such as BS Contact have already added support to Nvidia Physx, it would be quite possible to further extend current framework to enable physics engine-based visual and haptic interaction.

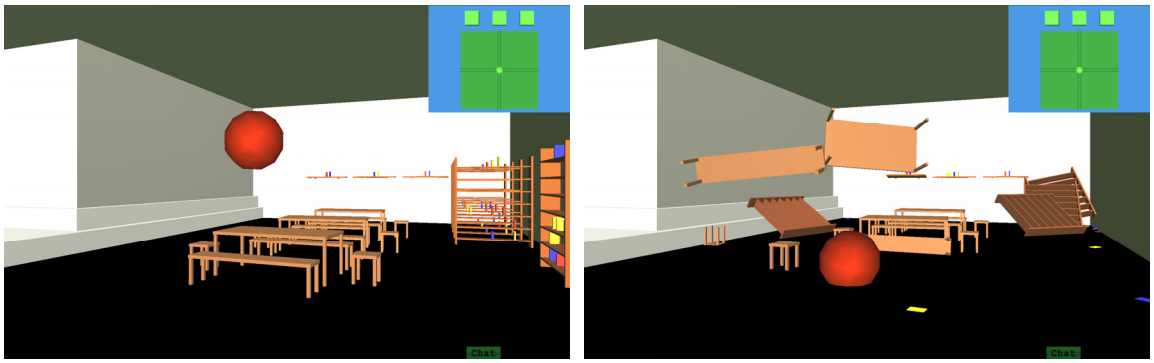


Figure 8.1 Feasibility study of integrating haptic interaction with physics engines

We have already accomplished a feasibility study in this research direction and implemented a few demos to validate the research novelty. In Figure 8.1, the scene is built based on our framework. All objects are declared to support Nvidia Physx. A haptic device is used to control the avatar (red sphere) in the scene to touch other objects. The objects will move when collision happens between them and the tool avatar. The actual force vector rendered on the haptic device is calculated based on our algorithm implemented in this thesis, while the opposite force vector (fake force) is used for

animating object movements are calculated by the physics engine.

8.2.2 Expansion of the uniform modeling paradigm with other human sensations

The five classic human sensations in the physical world: vision, hearing, taste, smell, and touch, are complementary. Among the five sensations, vision on computer is undoubtedly in dominance. In this thesis we proposed a uniform modeling paradigm to incorporate both vision and touch; however the uniform paradigm could be further expanded to support other classic human sensations. Certain research efforts have been made on this topic, such as haptics with smell [140, 141] and haptics with hearing [142, 143]; however our framework could support such expansion with better uniformity and efficiency. Ideally, various devices for human sensations such fragrance dispenser [144, 145], microphones and earphones could cooperate with visual devices (monitor, head mounted device) and haptic devices to build immersive virtual environments. Mathematical functions and procedures could be used on each and every sensation definition for accurate and compact representation.

8.2.3 Image and video-based haptic interaction and collaboration

Haptic devices are naturally designed for 3D applications. However, collision detection with 3D polygon-based and voxel-based models still requires intensive computation and therefore real time haptic interaction with complicated 3D models is still a big challenge. Since all 3D models will finally be projected to 2D screen, there will be no visual difference if 2D images and videos are used for guiding haptic interaction, while such

replacement will greatly reduce the computation resources needed for visual rendering and therefore accelerate haptic rendering. Based on this idea, image and video-based haptic interaction and collaboration could be proposed by extending current haptic collision detection and force rendering algorithms. Physical properties could be defined either by the information provided within the image and video or by augmented physical properties which will be superimposed on the images and videos. Mathematical functions and procedures could also be integrated to further accelerate haptic interaction and collaboration with complicated physical property definitions.

References

[1] I. Brent Edward, *Passive haptics significantly enhances virtual environments*, The University of North Carolina at Chapel Hill, 2001, pp. 1-100.
DOI: 10.1.1.17.3356
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.3356

[2] G. Singh, *Haptic Painting*, *Computer Graphics and Applications, IEEE,* 24 (2004), pp. 4-5.
DOI: 10.1109/MCG.2004.52
http://www.computer.org/portal/web/csdl/doi/10.1109/MCG.2004.52

[3] D. Johnson, T. V. Thompson, II, M. Kaplan, D. Nelson and E. Cohen, *Painting textures with a haptic interface*, *Virtual Reality, 1999. Proceedings., IEEE*, 1999, pp. 282-285.
DOI: 10.1109/VR.1999.756963
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isNumber=16390&arNumber=756963&isnumber=16390&arnumber=756963

[4] J. Hua and H. Qin, *Dynamic implicit solids with constraints for haptic sculpting*, *Shape Modeling International, 2002. Proceedings*, 2002, pp. 119-129.
DOI: 10.1109/SMI.2002.10010
http://www.computer.org/portal/web/csdl/doi/10.1109/SMI.2002.10010

[5] W. Zhu and Y.-S. Lee, *Product prototyping and manufacturing planning with 5-DOF haptic sculpting and dexel volume updating*, *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on*, 2004, pp. 98-105.
DOI: 10.1109/HAPTIC.2004.1287183
http://portal.acm.org/citation.cfm?id=1880285

[6] T. Coles, D. Meglan and N. John, *The Role of Haptics in Medical Training Simulators: A Survey of the State-of-the-art*, *Haptics, IEEE Transactions on,* (2010), pp. 1.
DOI: 10.1109/TOH.2010.19
http://doi.ieeecomputersociety.org/10.1109/TOH.2010.19

[7] S. Paneels and J. C. Roberts, *Review of Designs for Haptic Data Visualization*, *Haptics, IEEE Transactions on,* 3 (2010), pp. 119-137.
DOI: 10.1109/TOH.2009.44
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5255235

[8] D. J. Reinkensmeyer, C. T. Pang, J. A. Nessler and C. C. Painter, *Java Therapy: Web-Based Robotic Rehabilitation*, *Proceedings 7th. International Conference on Rehabilitation Robotics*, 2001, pp. 66-71.
DOI: 10.1.1.7.7930
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.7930

[9] I. Thouvenin, D. Lenne, A. Guenand and S. A. A. S. Aubry, *Knowledge integration in early design stages for collaboration on a virtual mock up*, in D. Lenne, ed., *Computer Supported Cooperative Work in Design, 2005. Proceedings of the Ninth International Conference on*, 2005, pp. 1141-1145 vol. 2.
DOI: 10.1109/CSCWD.2005.194350

http://www.computer.org/portal/web/csdl/doi/10.1109/CSCWD.2005.194350

[10] S. Aubry, I. Thouvenin, D. Lenne and J. Olive, *A knowledge model to read 3D annotations on a virtual mock-up for collaborative design*, in I. Thouvenin, ed., *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, 2007, pp. 669-674.
DOI: 10.1109/CSCWD.2007.4281516
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4281516

[11] L. Chen, Z. Song and L. Feng, *Internet-enabled real-time collaborative assembly modeling via an e-Assembly system: status and promise*, *Computer-Aided Design*, 2004, pp. 835-847.
DOI: 10.1016/j.cad.2003.09.010
http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6TYR-49NPTYT-2&_user=10&_coverDate=08%2F31%2F2004&_alid=1600758768&_rdoc=1&_fmt=high&_orig=search&_origin=search&_zone=rslt_list_item&_cdi=5625&_docanchor=&view=c&_ct=101&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=aee3730f23764799e7239b3c2ea673df&searchtype=a

[12] R. Iglesias, E. Prada, A. Uribe, A. Garcia-Alonso, S. Casado and T. Gutierrez, *Assembly Simulation on Collaborative Haptic Virtual Environments*, *Proceedings of the 15-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2007, pp. 241-247.
http://wscg.zcu.cz/wscg2007/Papers_2007/full/E23-full.pdf

[13] G. Q. Huang, K. L. Mak, J. Y. Shen and J. Q. A. Y. J. Q. Yan, *Web-based product design review: implementation perspective*, in K. L. Mak, ed., *Computer Supported Cooperative Work in Design, The Sixth International Conference on, 2001*, 2001, pp. 261-266.
DOI: 10.1109/CSCWD.2001.942269
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=942269

[14] R. Sidharta, J. Oliver and A. Sannier, *Augmented Reality Tangible Interface for Distributed Design Review*, in J. Oliver, ed., *Computer Graphics, Imaging and Visualisation, 2006 International Conference on*, 2006, pp. 464-470.
DOI: 10.1109/CGIV.2006.25
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1663834

[15] X. Wang and P. S. Dunston, *Usability Evaluation of a Mixed Reality Collaborative Tool for Design Review*, in P. S. Dunston, ed., *Computer Graphics, Imaging and Visualisation, 2006 International Conference on*, 2006, pp. 448-451.
DOI: 10.1109/CGIV.2006.87
http://www.computer.org/portal/web/csdl/doi/10.1109/CGIV.2006.87

[16] P. Buttolo and B. Hannaford, *Pen-based force display for precision manipulation in virtual environments*, in B. Hannaford, ed., *Virtual Reality Annual International Symposium, 1995. Proceedings.*, 1995, pp. 217-224.
DOI: 10.1109/VRAIS.1995.512499
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=512499

[17] A. Heuser, H. Kourtev, V. Hentz and P. A. F. P. Forducey, *Tele-Rehabilitation using the Rutgers Master II glove following Carpal Tunnel Release surgery*, in H. Kourtev, ed., *Virtual Rehabilitation, 2006 International Workshop on*, 2006, pp. 88-93.

DOI: 10.1109/IWVR.2006.1707533
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1707533

[18] S. Lee, S. Park, M. Kim and C.-W. Lee, *Design of a force reflecting master arm and master hand using pneumatic actuators*, in P. Sangmin, ed., *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 1998, pp. 2574-2579 vol.3.
DOI: 10.1109/ROBOT.1998.680729
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=680729

[19] http://www.mindflux.com.au/products/vti/cybergrasp.html.

[20] S. Iqbal Omar, A. Hernandez, R. McLauchlan and R. Challoo, *An approach for the teleoperator control of the Stanford/JPLDexterous Hand*, *Proceedings of Systems, Man, and Cybernetics. 'Computational Cybernetics and Simulation'*. 1997, pp. 4303-4308.
DOI: 10.1109/ICSMC.1997.637376
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=637376

[21] H. Seraji and R. Steele, *Nonlinear contact control for space station dexterous arms*, in R. Steele, ed., *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 1998, pp. 899-906 vol.1.
DOI: 10.1109/ROBOT.1998.677102
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=677102

[22] http://bleex.me.berkeley.edu/haptic.htm.

[23] P. Sines and B. Das, *Peltier Haptic Interface (PHI) for improved sensation of touch in virtual environments*, *Journal of Virtual Reality,* 4 (1999), pp. 260-264.
DOI: 10.1007/BF01421809
http://www.springerlink.com/content/v365jm23n2007557

[24] U. Kühnapfel, H. Cakmak and H. Maass, *Endoscopic surgery training using virtual reality and deformable tissue simulation*, *Computers & Graphics, Elsevier,* 24 (2000), pp. 671-682.
DOI: 10.1016/S0097-8493(00)00070-4
http://www.ingentaconnect.com/content/els/00978493/2000/00000024/00000005/art00070

[25] M. Akamatsu and S. Sato, *A multi-modal mouse with tactile and force feedback*, International Journal of Human-Computer Studies, 40 (1994), pp. 443-453.
http://www.sciencedirect.com/science/article/B6WGR-45NK0XW-C/2/dbc96dd55be8acbfc3549c00e081acf8

[26] J. Raisamo, R. Raisamo and K. Kosonen, *Distinguishing Vibrotactile Effects with Tactile Mouse and Trackball*, in T. McEwan, J. Gulliksen and D. Benyon, eds., *People and Computers XIX — The Bigger Picture*, Springer London, 2006, pp. 337-348.
DOI: 10.1007/1-84628-249-7_21
http://www.springerlink.com/content/t8r108x50h325397

[27] http://www.sensable.com.

[28] http://www.sensable.com/support-ghost-sdk.htm.

[29] http://www.sensable.com/products-openhaptics-toolkit.htm.

[30] http://www.immersion.com.

[31] http://www.forcedimension.com.

[32] http://www.chai3d.org.
[33] http://www.quanser.com.
[34] http://www.mpb-technologies.ca/.
[35] http://www-acroe.imag.fr/ergos-technologies/.
[36] http://home.novint.com.
[37] http://home.novint.com/products/sdk.php.
[38] E. Schömer, J. Reichel and T. Warken, *Efficient Collision Detection for Curved Solid Objects*, *Proceedings of the seventh ACM symposium on Solid modeling and applications*, ACM, Saarbrücken, Germany, 2002, pp. 321-328.
DOI: 10.1145/566282.566328.
http://portal.acm.org/citation.cfm?id=566328
[39] N. R. El-Far, N. D. Georganas and A. El-Saddik, *A Collision Detection Algorithm for Point-Like Haptic Interactions in Highly Detailed Virtual Environments*, *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2007. VECIMS 2007. IEEE Symposium on*, 2007, pp. 25-30.
DOI: 10.1109/VECIMS.2007.4373922
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4373922
[40] M. Brian, *V-Clip: fast and robust polyhedral collision detection*, ACM Trans. Graph., 17 (1998), pp. 177-208.
DOI: 10.1145/285857.285860
http://portal.acm.org/citation.cfm?id=285860
[41] K. Moustakas, D. Tzovaras and M. G. Strintzis, *SQ-Map: Efficient Layered Collision Detection and Haptic Rendering*, *Visualization and Computer Graphics, IEEE Transactions on,* 13 (2007), pp. 80-93.
DOI: 10.1109/TVCG.2007.20
http://www.computer.org/portal/web/csdl/doi/10.1109/TVCG.2007.20
[42] C. Raymaekers, K. Beets and F. V. Reeth, *Fast haptic rendering of complex objects using subdivision surfaces*, *Proceedings of the Sixth PHANToM Users Group Workshop*, 2001, pp. 19-22.
http://www.cs.kent.edu/~yhijazi/mesh/haptic%20face/FastHaptic.pdf
[43] K. Laehyun, A. Kyrikou, G. S. Sukhatme and M. Desbrun, *An implicit-based haptic rendering technique*, *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 2002, pp. 2943-2948 vol.3.
DOI: 10.1109/IRDS.2002.1041719
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1041719
[44] D. C. Ruspini, K. Kolarov and O. Khatib, *The haptic display of complex graphical environments*, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 345-352.
DOI: 10.1145/258734.258878
http://portal.acm.org/citation.cfm?id=258878
[45] A. Gregory, M. C. Lin, S. Gottschalk and R. Taylor, *A framework for fast and accurate collision detection for haptic interaction*, *ACM SIGGRAPH 2005 Courses*, ACM, Los Angeles, California, 2005, pp. 38-45.
DOI: 10.1145/1198555.1198604
http://portal.acm.org/citation.cfm?id=1198604

[46] L. S. H. Chan and C. Kup-Sze, *Integrating PhysX and OpenHaptics: Efficient force feedback generation using physics engine and haptic devices*, *Pervasive Computing (JCPC), 2009 Joint Conferences on*, 2009, pp. 853-858.
DOI: 10.1109/JCPC.2009.5420068
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5420068
[47] http://www.nvidia.com/object/physx_new.html.
[48] http://www.nvidia.com/object/cuda_home_new.html.
[49] http://www.havok.com/index.php?page=havok-physics.
[50] http://www.ode.org/.
[51] http://bulletphysics.org/wordpress/.
[52] http://www.newtondynamics.com.
[53] A. Sourin and L. Wei, *Visual immersive haptic rendering on the web*, *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, ACM, Singapore, 2008.
DOI: 10.1145/1477862.1477890
http://doi.acm.org/10.1145/1477862.1477890.
[54] C. B. Zilles and J. K. Salisbury, *A constraint-based god-object method for haptic display*, *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 1995, pp. 146-151 vol.3.
DOI: 10.1109/IROS.1995.525876
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=525876
[55] G. C. Burdea, *Force and touch feedback for virtual reality*, John Wiley & Sons, Inc., New York, NY, USA, 1996
[56] S. Booth, F. D. Angelis and T. Schmidt-Tjarksen, *The Influence of Changing Haptic Refresh-Rate on Subjective User Experiences - Lessons for Effective Touch-Based Applications*, *EuroHaptics*, Dublin, Ireland 2003, pp. 374-383.
http://www.eurohaptics.vision.ee.ethz.ch/2003/13.pdf
[57] R. Weller and G. Zachmann, *A unified approach for physically-based simulations and haptic rendering*, *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, ACM, New Orleans, Louisiana, 2009, pp. 151-159.
DOI: 10.1145/1581073.1581097
http://portal.acm.org/citation.cfm?id=1581097
[58] N. R. El-Far, N. D. Georganas and A. E. Saddik, *Collision Detection and Force Response in Highly-Detailed Point-Based Hapto-Visual Virtual Environments*, *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications, IEEE Computer Society,* 2007, pp. 15-22.
DOI: 10.1109/DS-RT.2007.17
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4384525
[59] D. Knott and D. K. Pai, *CInDeR : collision and interference detection in real time using graphics hardware*, *Proceedings of Graphics Interface*, 2003, pp. 73-80.
DOI: 10.1.1.10.316
http://www.cs.rutgers.edu/~dpai/papers/KnottPai03.pdf
[60] P. Kolčárek and J. Sochor, *Velocity driven haptic rendering*, *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM, Dunedin, New Zealand, 2005, pp.

389-394.
DOI: 10.1145/1101389.1101465
http://portal.acm.org/citation.cfm?id=1101465

[61] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin and D. Manocha, *Six degree-of-freedom haptic display of polygonal models*, *Proceedings of the conference on Visualization '00*, IEEE Computer Society Press, Salt Lake City, Utah, United States, 2000, pp. 139-146.
DOI: 10.1109/VISUAL.2000.885687
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isNumber=19150&arNumber=885687&isnumber=19150&arnumber=885687

[62] W. A. McNeely, K. D. Puterbaugh and J. J. Troy, *Advances in voxel-based 6-DOF haptic rendering*, *ACM SIGGRAPH 2005 Courses*, ACM Press, Los Angeles, California, 2005.
DOI: 10.1145/1198555.1198606
http://portal.acm.org/citation.cfm?id=1198606

[63] J. Hua and H. Qin, *Haptics-based volumetric modeling using dynamic spline-based implicit functions*, *Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, IEEE Press, Boston, Massachusetts, 2002, pp. 55-64.
DOI: 10.1109/SWG.2002.1226510
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isNumber=27525&arNumber=1226510&isnumber=27525&arnumber=1226510

[64] S.-Y. Kim, J. Park and D.-S. Kwon, *Multiple-contact representation for the real-time volume haptic rendering of a non-rigid object*, in P. Jinah, ed., *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on*, 2004, pp. 242-249.
DOI: 10.1109/HAPTIC.2004.1287202
http://portal.acm.org/citation.cfm?id=1880305

[65] K. Lundin, B. Gudmundsson and A. Ynnerman, *General Proxy-based Haptics for Volume Visualization*, *Proceedings of the World Haptics Conference*, Italy, 2005, pp. 557-560.
DOI: 10.1109/WHC.2005.62
http://www.computer.org/portal/web/csdl/doi/10.1109/WHC.2005.62

[66] K. Lundin, C. Lundström, M. Cooper and A. Ynnerman, *Enabling Haptic Interaction with Volumetric MRI Data Through Knowledge-based Tissue Separation*, *Proceedings of Volume Graphics 2006*, 2006, pp. 75-78.
DOI: 10.1.1.96.1961
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.1961

[67] http://www.sensegraphics.com/index.php?option=com_content&task=view&id=115&Itemid=83.

[68] N. W. John, M. Riding, N. I. Phillips, S. Mackay, L. Steineke, B. Fontaine, G. Reitmayr, V. Valencic, N. Zimic, A. Emmen, E. Manolakaki and D. Theodoros, *Web-based surgical educational tools*, Studies in health technology and informatics (2001), pp. 212-217.
http://www.ncbi.nlm.nih.gov/pubmed/11317742

[69] G. Reitmayr, S. Carroll, A. Reitemeyer and M. G. Wagner, *DeepMatrix - An open technology based virtual environment system*, The Visual Computer (1999), pp. 395-412.
DOI: 10.1007/s003710050187
http://www.springerlink.com/content/q9bnr8m7hegl0jbt/
[70] http://www.onlive.com/.
[71] http://www.youtube.com/watch?v=zpFzpF0msrU.
[72] A. Chong, A. Sourin and K. Levinski, *Grid-based computer animation rendering*, *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, ACM, Kuala Lumpur, Malaysia, 2006, pp. 39-47.
DOI: 10.1145/1174429.1174435
http://portal.acm.org/citation.cfm?id=1174435
[73] B.-Y. Chen and T. Nishita, *Multiresolution streaming mesh with shape preserving and QoS-like controlling*, *Proceedings of the seventh international conference on 3D Web technology*, ACM, Tempe, Arizona, USA, 2002, pp. 35-42.
DOI: 10.1145/504502.504509
http://portal.acm.org/citation.cfm?id=504509
[74] S. Rusinkiewicz and M. Levoy, *Streaming QSplat: a viewer for networked visualization of large, dense models*, *Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM, 2001, pp. 63-68.
DOI: 10.1145/364338.364350
http://portal.acm.org/citation.cfm?doid=364338.364350
[75] M. Deering, *Geometry compression*, *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, pp. 13-20.
DOI: 10.1145/218380.218391
http://portal.acm.org/citation.cfm?id=218391
[76] A. Khodakovsky, P. Schröder and W. Sweldens, *Progressive geometry compression*, *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 271-278.
DOI: 10.1145/344779.344922
http://portal.acm.org/citation.cfm?id=344922
[77] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo and R. Bajcsy, *Enabling multi-party 3D tele-immersive environments with ViewCast*, *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP),* 6 (2010), pp. 1-30.
DOI: 10.1145/1671962.1671963
http://portal.acm.org/citation.cfm?id=1671962.1671963
[78] http://www.web3d.org/x3d/specifications/vrml.
[79] http://www.web3d.org/x3d/specifications/#x3d.
[80] http://www.collada.org.
[81] http://www.opencroquet.org/index.php/Main_Page.
[82] http://www.opencobalt.org/.
[83] http://www.secondlife.com.
[84] http://www.activeworlds.com.
[85]

http://www.informationweek.com/news/software/hosted/showArticle.jhtml?articleID=197800179.

[86] C. Bajaj and S. Cutchin, *Web based collaborative visualization of distributed and parallel simulation*, *Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics*, San Francisco, California, United States, 1999, pp. 47-54.
DOI: 10.1109/PVGS.1999.810138
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=810138

[87] G. Konduri and A. Chandrakasan, *A framework for collaborative and distributed web-based design*, *Proceedings of the 36th ACM/IEEE conference on Design automation*, ACM Press, New Orleans, Louisiana, United States, 1999, pp. 898-903.
DOI: 10.1109/DAC.1999.782214
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=782214

[88] J. W. Chastine, J. C. Brooks, Y. Zhu, G. S. Owen, R. W. Harrison and I. T. Weber, *AMMP-Vis: a collaborative virtual environment for molecular modeling*, *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, Monterey, CA, USA, 2005, pp. 8-15.
DOI: 10.1145/1101616.1101620
http://portal.acm.org/citation.cfm?id=1101620

[89] M. Okada, H. Tarumi and T. Yoshimura, *Distributed virtual environment realizing collaborative environment education*, *Proceedings of the 2001 ACM symposium on Applied computing*, ACM Press, Las Vegas, Nevada, United States, 2001, pp. 83-88.
DOI: 10.1145/372202.372281
http://portal.acm.org/citation.cfm?doid=372202.372281

[90] R. Halvorsrud and S. Hagen, *Designing a collaborative virtual environment for introducing pupils to complex subject matter*, *Proceedings of the third Nordic conference on Human-computer interaction*, ACM Press, Tampere, Finland, 2004, pp. 121-130.
DOI: 10.1145/1028014.1028034
http://portal.acm.org/citation.cfm?id=1028014.1028034

[91] M. Tang, S.-C. Chou and J.-X. Dong, *Collaborative virtual environment for feature based modeling*, *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, ACM, Singapore, 2004, pp. 120-126.
DOI: 10.1145/1044588.1044610
http://portal.acm.org/citation.cfm?id=1044610

[92] G. Alma Martínez, A. Héctor Orozco, C. Félix Ramos and M. Siller, *A Peer-to-Peer Architecture for Real-Time Distributed Visualization of 3D Collaborative Virtual Environments*, *Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, IEEE Computer Society, 2009, pp. 251-254.
DOI: 10.1109/DS-RT.2009.21
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5361755

[93] A. V. Mironova, *Collaborative Volume Visualization Using VTK*, *Proceedings of*

the 14th IEEE Visualization 2003 (VIS'03), IEEE Computer Society, 2003, pp. 99.
DOI: 10.1109/VIS.2003.10015
http://www.computer.org/portal/web/csdl/doi/10.1109/VIS.2003.10015

[94] P. Bourne, M. Gribskov, G. Johnson, J. Moreland, S. Wavra and H. Weissig, *A prototype molecular interactive collaborative environment*, *Proceedings of the Pacific Symposium on Biocomputing*, 1998, pp. 118--129.
DOI: 10.1.1.47.7473
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.7473

[95] J. Wang, S. Zhang and H. Sun, *VRML based collaborative visualization for volume product datasets*, *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, 2004, pp. 94-98 vol.1.
DOI: 10.1109/CACWD.2004.1348995
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1348995

[96] B. Knoerlein, Gabor Szekely and M. Harders, *Visuo-haptic collaborative augmented reality ping-pong*, *Proceedings of the international conference on Advances in computer entertainment technology*, ACM Press, Salzburg, Austria, 2007.
DOI: 10.1145/1255047.1255065
http://portal.acm.org/citation.cfm?id=1255065

[97] A. A. Pasko, V. Adzhiev, A. Sourin and V. V. Savchenko, *Function representation in geometric modeling: concepts, implementation and applications*, The Visual Computer, Springer, 11 (1995), pp. 429-446.
DOI: 10.1007/BF02464333
http://www.springerlink.com/content/r516631882773282/

[98] http://www.h3d.org.

[99] M. O'Malley and S. Hughes, *Simplified authoring of 3D haptic content for the World Wide Web*, in S. Hughes, ed., *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on*, 2003, pp. 428-429.
DOI: 10.1109/HAPTIC.2003.1191334
http://www.computer.org/portal/web/csdl/doi/10.1109/HAPTIC.2003.1191334

[100] T. Asano, Y. Ishibashi, S. Minezawa and M. Fujimoto, *Surveys of exhibition planners and visitors about a distributed haptic museum*, *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACM, Valencia, Spain, 2005, pp. 246-249.
DOI: 10.1145/1178477.1178518
http://portal.acm.org/citation.cfm?id=1178518

[101] M. McLaughlin, G. Sukhatme and C. Shahabi, *The Haptic Museum*, *Proceedings of the EVA 2000 Conferences on Electronic Imaging and the Visual Arts*, 2000.
http://infolab.usc.edu/DocsDemos/eva2000.pdf

[102] http://www.hapticweb.org.

[103] E. Ruffaldi, A. Frisoli, M. Bergamasco, C. Gottlieb and F. Tecchia, *A haptic toolkit for the development of immersive and web-enabled games*, *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, Limassol, Cyprus, 2006, pp. 320-323.
DOI: 10.1145/1180495.1180559

http://portal.acm.org/citation.cfm?id=1180559
[104] http://www.vrmedia.it.
[105] A. Hamam, S. Nourian, N. R. El-Far, F. Malric, X. Shen and N. D. Georganas, *A Distributed, Collaborative and Haptic-Enabled Eye Cataract Surgery*, *Proceedings of IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2006, pp. 105-110.
DOI: 10.1109/HAVE.2006.283773
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4062520
[106] X. Shen, J. Zhou, A. E. Saddik and N. D. Georganas, *Architecture and Evaluation of Tele-Haptic Environments*, *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real Time Applications*, Hungary, 2004, pp. 53-60.
DOI: 10.1109/DS-RT.2004.8
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1364579
[107] C. Basdogan, C.-H. Ho, M. ASrinavasan and M. Slater, *An Experimental Study on the Role of Touch in Shared Virtual Environments*, *Proceedings of ACM Transactions on Computer-Human Interaction*, 2000, pp. 443-460.
DOI: 10.1145/365058.365082
http://portal.acm.org/citation.cfm?id=365082
[108] D. Morris, N. Joshi and K. Salisbury, *Haptic Battle Pong: High-Degree-of-Freedom Haptics in a Multiplayer Gaming Environment*, *Experimental Gameplay Workshop, GDC*, 2004.
DOI: 10.1.1.84.533
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.533
[109] I. Goncharenko, M. Svinin, S. Matsumoto, Y. Masui, Y. Kanou and S. Hosoe, *Cooperative Control with Haptic Visualization in Shared Virtual Environments*, *Proceedings of 8th. Int Conf on Information Visualization, IV04*, 2004, pp. 533-538.
DOI: 10.1109/IV.2004.1320196
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1320196
[110] I. Fukuda and S. Matsumoto, *A Robust System for Haptic Collaboration over the Network*, *Proc Touch in Virtual Environments Conference*, 2002.
[111] K. Hikichi, I. Arimoto, H. Morino, K. Sezaki and Y. Yasuda, *Evaluation of Adaptation Control for Haptics Collaboration over the Internet*, *Proceedings of IEEE Communications Quality & Reliability Int Workshop*, 2002, pp. 218-222.
[112] M. Alhalabi and S. Horiguchi, *Tele-Handshake: A Cooperative Shared Haptic Virtual Environment*, *Proceedings of EuroHaptics*, UK, 2001, pp. 60-64.
DOI: 10.1.1.2.6840
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.6840
[113] J. Zhou, X. Shen and N. D. Georganas, *Haptic Tele-Surgery Simulation*, *Proceedings of the 3rd IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications (HAVE' 2004)*, 2004, pp. 99-104.
DOI: 10.1109/HAVE.2004.1391889
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1391889
[114] M. Y. Sung, Y. Yoo, K. Jun, N.-J. Kim and J. Chae, *Experiments for a Collaborative Haptic Virtual Reality*, *Proceedings of the 16th International*

Conference on Artificial Reality and Telexistence--Workshops (ICAT'06), 2006, pp. 174-179.
DOI: 10.1109/ICAT.2006.60
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4089234

[115] R. J. Hubbold, *Collaborative stretcher carrying: a case study*, *Proceedings of the workshop on Virtual environments 2002*, Eurographics Association, Barcelona, Spain, 2002, pp. 7-12.
http://portal.acm.org/citation.cfm?id=509711

[116] J. Caroline, G. Mashhuda and H. Roger, *Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment*, ACM Trans. Comput.-Hum. Interact., 14 (2007).
DOI: 10.1145/1275511.1275514
http://portal.acm.org/citation.cfm?id=1275514

[117] I. Vaghi, C. Greenhalgh and S. Benford, *Coping with inconsistency due to network delays in collaborative virtual environments*, *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, London, United Kingdom, 1999, pp. 42-49.
DOI: 10.1145/323663.323670
http://portal.acm.org/citation.cfm?id=323670

[118] Q. Liu and A. Sourin, *Function-defined shape metamorphoses in visual cyberworlds*, *The Visual Computer, Springer,* 2006, 22 (2006), pp. 977-990.
DOI: 10.1007/s00371-006-0044-0
http://www.springerlink.com/content/44u1h415j1v42514/

[119] M. Arbabtafti, M. Moghaddam, A. Nahvi, M. Mahvash and A. Rahimi, *Haptic and visual rendering of virtual bone surgery: A physically realistic voxel-based approach*, *Haptic Audio visual Environments and Games, 2008. HAVE 2008. IEEE International Workshop on*, 2008, pp. 30-35.
DOI: 10.1109/HAVE.2008.4685294
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4685294

[120] M. Arbabtafti, M. Moghaddam, A. Nahvi, M. Mahvash, B. Richardson and B. Shirinzadeh, *Physics-Based Haptic Simulation of Bone Machining*, *Haptics, IEEE Transactions on,* (2010), pp. 1.
DOI: 10.1109/TOH.2010.5
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5406525

[121] D. Morris, C. Sewell, F. Barbagli, K. Salisbury, N. H. Blevins and S. Girod, *Visuohaptic Simulation of Bone Surgery for Training and Evaluation*, *Computer Graphics and Applications, IEEE,* 26 (2006), pp. 48-57.
DOI: 10.1109/MCG.2006.140
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4012565

[122] Z. Lu, G. Sankaranarayanan, D. Deo, D. Chen and S. De, *Towards physics-based interactive simulation of electrocautery procedures using PhysX*, *Haptics Symposium, 2010 IEEE*, 2010, pp. 515-518.
DOI: 10.1109/HAPTIC.2010.5444609
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5444609

[123] V. Daniulaitis, M. O. Alhalabi, H. Kawasaki and Y. Tanaka, *Medical palpation of deformable tissue using physics-based model for haptic interface robot (HIRO)*,

Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, 2004, pp. 3907-3911 vol.4.
DOI: 10.1109/IROS.2004.1390024
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1390024

[124] C. Basdogan, C.-H. Ho and M. A. Srinivasan, *A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments*, *Proceedings of the ASME Dynamic Systems and Control Division*, 1997, pp. 77-84.
http://www.rle.mit.edu/touchlab/publications/1997_001.pdf

[125] K. Lundin, A. Ynnerman and B. Gudmundsson, *Proxy-based Haptic Feedback from Volumetric Density Data*, *Euro Haptics*, 2002, pp. 104-109.
DOI: 10.1.1.5.5680
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.5680

[126] R. Weller and G. Zachmann, *Inner Sphere Trees and Their Application to Collision Detection*, Virtual Realities (2011), pp. 181-201.
DOI: 10.1007/978-3-211-99178-7_10
http://www.springerlink.com/content/g20x1797h2p157q4/

[127] Y. Sheng, A. Sourin, G. G. Castro and H. Ugail, *A PDE method for patchwise approximation of large polygon meshes*, The Visual Computer, 26 (2010), pp. 975-984.
DOI: 10.1007/s00371-010-0456-8
http://www.springerlink.com/content/e7287jp57723n3h6/

[128] F. Lai and A. Sourin, *Function-defined shape node for VRML*, *Eurographics 2002, Short Presentations, Eurographics Press*, 2002, pp. 207-215.
http://cospace.sce.ntu.edu.sg/Shared/Sourin/Papers/short19.pdf

[129] Q. Liu and A. Sourin, *Function-based representation of complex geometry and appearance*, *Proceedings of the tenth international conference on 3D Web technology*, ACM, Bangor, United Kingdom, 2005, pp. 123-134.
DOI: 10.1145/1050491.1050509
http://portal.acm.org/citation.cfm?id=1050509

[130] Q. Liu and A. Sourin, *Function-based shape modeling and visualization in X3D*, *Proceedings of the eleventh international conference on 3D web technology*, ACM, Columbia, Maryland, 2006, pp. 131-141.
DOI: 10.1145/1122591.1122609
http://portal.acm.org/citation.cfm?id=1122609

[131] Q. Liu and A. Sourin, *Function-based shape modelling extension of the Virtual Reality Modelling Language*, *Computers & Graphics, Elsevier,* 2006, 30 (4), pp. 629-645.
DOI: 10.1.1.134.375
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.375

[132] Q. Liu and A. Sourin, *Function-defined shape metamorphoses in visual cyberworlds*, *The Visual Computer, Springer,* 2006, 22 (12), pp. 977-990.
DOI: 10.1007/s00371-006-0044-0
http://www.springerlink.com/content/44u1h415j1v42514/

[133] http://www.deepmatrix.org.

[134] http://www.blaxxun.com.

[135] http://kimballsoftware.com/abnet.
[136] http://www.planet9.com/products.html.
[137] http://www.octaga.com/joomla/index.php.
[138] http://www.bitmanagement.com.
[139] D. Brutzman and L. Daly, eds., *X3D: 3D Graphics for Web Authors*, Morgan Kaufmann, 2007.
[140] K. Miyahara and Y. Okada, *COLLADA-Based File Format Supporting Various Attributes of Realistic Objects for VR Applications*, *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on*, 2009, pp. 971-976.
DOI: 10.1109/CISIS.2009.161
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5066909
[141] B. S. Spencer, *Incorporating the sense of smell into patient and haptic surgical simulators*, *Information Technology in Biomedicine, IEEE Transactions on,* 10 (2006), pp. 168-173.
DOI: 10.1109/TITB.2005.856851
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1573718
[142] P. Roth, D. Richoz, L. Petrucci and T. Pun, *An audio-haptic tool for non-visual image representation*, *Signal Processing and its Applications, Sixth International, Symposium on. 2001*, 2001, pp. 64-67 vol.1.
DOI: 10.1109/ISSPA.2001.949776
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=949776
[143] B. Menelas, L. Picinalli, B. F. G. Katz and P. Bourdot, *Audio haptic feedbacks for an acquisition task in a multi-target context*, *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, 2010, pp. 51-54.
DOI: 10.1109/3DUI.2010.5444722
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5444722
[144] www.engadget.com/2006/04/12/japanese-movie-theaters-to-get-internet-controlled-smell-o-visi.
[145] www.engadget.com/2006/09/06/forget-smell-o-vision-usb-aroma-geur-lets-you-smell-the-radio.

Appendix List of Literature

Journal Papers

1. **Lei Wei**, Alexei Sourin, Olga Sourina, "Function-based Visualization and Haptic Rendering in Shared Virtual Spaces", The Visual Computer, Springer, 24(10):871-880, DOI: 10.1007/s00371-008-0285-1, 2008.
2. Alexei Sourin, Olga Sourina, **Lei Wei**, Paul Gagnon, "Visual Immersive Haptic Mathematics in Shared Virtual Spaces", Transactions on Computational Science III, LNCS5300, Springer, LNCS Volume 5300/2009, pp 1-19, 2009.
3. Alexei Sourin, **Lei Wei**, "Visual Immersive Haptic Mathematics", Virtual Reality, Springer, DOI 10.1007/s10055-009-0133-2, 2009.
4. **Lei Wei**, Alexei Sourin, Olga Sourina, "Visualization and Haptic Rendering of Virtual Objects Defined by Mathematical Functions", Computer Graphics & Geometry, MEPhl, Russia, 11(1): 29-43, Spring 2009.
5. **Lei Wei**, Alexei Sourin, "Function-based Approach to Mixed Haptic Effects Rendering", accepted for publication by The Visual Computer, Springer. DOI: 10.1007/s00371-011-0548-0, 2011.

Conference Papers

1. **Lei Wei**, Alexei Sourin, Olga Sourina, "Function-based haptic interaction in Cyberworlds", in Proceedings of 2007 International Conference on Cyberworlds (CW 07), 24-27 Oct, 2007, Hannover, Germany. pp. 225-232.
2. Alexei Sourin, **Lei Wei**, "Visual Immersive Haptic Rendering on the Web", in Proceedings of the 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI 2008), 8-9 Dec, 2008, Singapore.
3. **Lei Wei**, Alexei Sourin, Herbert Stocker, "Function-based haptic collaboration in X3D", in Proceedings of Web3D 2009, 16-17 June, 2009, Darmstadt, Germany. pp. 15-23.
4. **Lei Wei**, Alexei Sourin, Herbert Stocker, "Collaboration in 3D Shared Spaces using X3D and VRML", in Proceedings of 2009 International Conference on Cyberworlds (CW 09), 7-11 Sept, 2009, Bradford, UK. pp. 36-42.
5. **Lei Wei**, Alexei Sourin, "Haptic Rendering of Mixed Haptic Effects", in Proceedings of 2010 International Conference on Cyberworlds (CW 10), 20-22 Oct, 2010, Singapore. pp. 38-45
6. **Lei Wei**, Alexei Sourin, Herbert Stocker, "A Framework for Visual and Haptic Collaboration in Shared Virtual Spaces", in Proceedings of the 6th International Symposium on Visual Computing (ISVC 2010), 27 Nov – 1 Dec, 2010, Las Vegas.