Distributed V.R. Training System for Endovascular Procedures

Nuha H. El-Khalili and Ken W. Brodlie

School of Computer Studies, University of Leeds,
Leeds LS2 9JT, United Kingdom

e-mail: *nuha@scs.leeds.ac.uk, kwb@scs.leeds.ac.uk*

The World Wide Web has grown to be a huge pool for exchanging information and services. Indeed it has become a framework for distributed computing applications, with user interface provided through a web browser. It has potential in particular for distributed simulation applications, providing world wide access to training systems. In this paper we will report results from our prototype web-based virtual reality simulation system which aims to train surgeons on endovascular procedures. The paper discusses features of the current technology that can support or limit the implementation of this type of application.

Keywords: Surgical simulator, VRML, Java, Web- based applications.

1. Introduction

Despite its short life time, the influence of the World Wide Web (WWW) has grown exponentially. The WWW has become a medium for exchanging information. Furthermore, there is an increasing awareness of its usefulness as a service provider medium. Thus, software developers are targeting the web as an environment for the next generation of applications. These applications will inherit from the web its world wide accessibility feature as well as facilitating multi-user access and collaborative working.

In this paper we present a prototype of a web-based virtual reality (VR) training simulation system for endovascular procedures. This work is part of a project carried out in conjunction with radiologists in St. James's University Hospital to build a training simulator for endovascular procedures used to treat abdominal aortic aneurysms (AAA). The paper will propose an architectural design for this type of application. We will also discuss the limitations in the current technology which hinders the achievement of some of the application requirements.

In the following section, we present our endovascular application, and we discuss the traditional method of solving this problem using a VR training simulator. In section three, the advantages of providing a web- based solution to this problem are presented. Section four presents the architectural design of our prototype and some implementation details. Section five will contain some concluding remarks; and finally reports the next stage of our research.

2. Endovascular Training Simulator

The need to build an endovascular training simulator arises from the development of new surgical techniques, specifically endoscopy. Endoscopy is a surgical technology which allows the operation on regions within the body with minimal damage to skin and surrounding tissues [Dumay 1995]. The advantages of endoscopy- being a minimally invasive technique- is that the patient’s discomfort and recovery time is reduced, consequently cost is reduced and health care standard is increased. However, this technology comes with its difficulties; these emerge from the ‘operation from a distance’ methodology. The radiologist performs the operation from a small incision far from the diseased organ using a long tubular tool called the endoscope. In addition, the radiologist is not able to directly view the operational area, but rather, vision is accessible via a video monitor. Alongside these difficulties are the inefficient current training methods. The one-to-one mentoring training method is time consuming and hazard prone. Meanwhile plastic models, animals and cadavers are expensive and inaccurate as a basis for training. This motivates the idea of a computer- based endovascular training simulator where radiologists can learn safely and relatively cheaply.

Simulation can be defined as the process of "modelling a system in such a way that the model mimics the response of the actual system to events that take place over time" [Pegden 1995]. Therefor, a simulator must respond to events over time, in a realistic manner. Thus, training simulators require interaction and realism.

In an endovascular training system realism would mean that the modelled anatomy and surgical tools must ‘look’ like the real objects. It would also mean that the models must ‘act’ like the real objects- e.g. the endoscope must bend when hitting the walls of the vessels. Furthermore, in endoscopic procedures the surgeon depends on the touch sensation to replace part of the lost vision. Thus, an additional requirement for realism is that the model ‘feels’ real. All these factors must be delivered in real time.

Virtual reality technology allows the generation of environments that look, feel, sound and even smell real. VR delivers this realistic environment by providing a closed loop of interaction with the modelled world via intuitive devices. Visual realism is facilitated by high performance rendering, and realistic dynamics is achieved by incorporating behaviour modelling.

These features of VR make it an attractive technology for training simulation systems in general, and surgical simulators in particular [Hon 1996]. Indeed there have been a number of attempts to build training simulator systems for endoscopy [CieMed, Meglan 1996, Walsum 1997].

These endoscopy simulators take advantage of VR technology to provide realistic training environments. However, they share some recognised deficiencies of conventional VR approaches. The difficult trade- off between visually appealing model representation and accuracy, and real- time interaction, drives these simulators to powerful dedicated hardware, and systems that are expensive and non- portable.

In this research, we are studying whether a simpler solution is feasible. Can we utilise the capabilities of the web in order to develop an endovascular simulator that can be used from a conventional browser?

3. Web- based Endovascular Training Simulator

There has been a growing interest in exploring the utility of the web environment in simulation applications. This interest can be classified into two main approaches. One approach is to perform the simulation task on a server and present the result on the client side. Attempts classified under this approach include the use of forms and CGI scripts to control the simulator parameters, and HTML pages for communication [Schumann 1997]. The second approach is to execute the whole simulation process on the client side, where Java applications and applets are utilised. Examples of this approach are the Java-based simulation libraries, such as simjava and simanim [Simjava].

As mentioned before, VR technology promises a lot of features but on the other hand makes high requirements. This seems to slow its growth as a technology for scientific applications. Hence, we need to change this restrictive paradigm of VR to a more open, accessible and scaleable one. This paradigm can be achieved if VR is migrated to the web environment. By building platform- independent VR applications and providing world wide access to them via the web, we would argue that this will create the necessary demand which can force VR technology to a more affordable and accessible technology and will spread its applications to a wider spectrum of fields.

There are already examples of desktop VR systems on the web- e.g. the virtual bicycle riding system by Bell Laboratories [Murray 1997]. These applications utilise the VRML- virtual reality modelling language- standard to implement their systems. At present VRML offers a standard for representing 3D dynamic worlds over the Internet. However, when VRML was introduced, it was thought to provide virtual reality over the Internet. There are several limitations in this standard that prevents it from achieving its ultimate goal. One of the major drawbacks in VRML that keeps its applications chained to a desktop configuration is its user interface which does not support immersive VR. Work done by [Edwards 1996] transfers part of the VRML interface from the browser to the virtual environment, which provides the flexibility of adding interface objects as needed and from different web locations.

By utilising VRML more in building VR applications on the web, we investigate its strengths and limitations with regard to this type of application- hence, getting it closer to its target of “VR on the web”.

4. Endovascular Prototype

Our aim is to develop a portable, scalable and ubiquitous VR simulator to help surgeons to acquire some of the skills needed to perform endovascular procedures which are used to treat AAA. Skills targeted by our simulator are eye/hand co-ordination, navigation through vessels, 3D perception and understanding of anatomical structures and decision making concerning the positioning and deployment of stents.

4.1 Architectural design

The endovascular simulator consists of four modules. The *simulator interface* is responsible for communicating actions and feedback from and to the user. Geometric representation and physical properties of the modelled objects are described in the *simulator database*. The *simulator engine* is responsible for creating time dependent events and synchronising the simulation processes. The *simulator processor* is responsible for responding to the system events based on the objects' properties specified in the database.

Because of the special requirements of a web- based VR system, there are two important issues that must be considered when designing this application: scalable architecture and portable implementation.

To achieve scalability in this simulator, we need to separate out modules which depend on special requirements that highly vary from one user environment to the other. For example, the input interface of a VR simulator may vary from a 2D mouse to a special 3D device with force feedback and position sensors. In this case, we need to create an interface between the input/ output module and the rest of the simulator modules, which will allow for the flexibility of changing the interface without the need to change the rest of the simulator.

The two general approaches for implementing simulation applications on the web do equally apply in the endovascular simulator case. One option- option A- is to implement the simulator such that it will execute on the client side, while the server role is merely a software provider. In this case all the system modules must be implemented in a platform independent language that is accessible through the web. Currently, the draft VRML 2.0 standard provides a good environment in which to represent 3D worlds and incorporate behavioural dynamics in it. VRML 2.0 would successfully be able to represent the database and the engine modules. It would also provide some standard form of simulator interface. Depending on the complexity of the simulator processor, it can either be implemented using VRML script nodes or as a

Java applet which runs alongside the VRML world. Because of the complex behaviour of the endoscope which requires numerical differential equations to represent it, this option is problematic under currently available technology. An established Java numerical library capable of solving ordinary differential equations is not yet available. In addition, the efficiency of Java for numerical calculations is still in question.

However, there are several efficient options of numerical libraries that are available in other platform dependent languages such as C and FORTRAN. Thus, to maintain our platform- independence feature, we need to restrict platform dependent modules away from the client side. This situation favours the second approach of web simulation configuration.

In this second approach- option B- the server acts as an application controller. The client will send simulation commands and parameters to the server. The actual simulation processing is performed on the server side. Results are then sent back to the client for display. This approach introduces a need to consider the network transfer rate and the synchronisation between the server and the client.

In our endovascular prototype, we have implemented the second approach. Figure (1) shows the data flow between the simulator modules under this approach. The next section will discuss some implementation details of the different modules of the simulator.

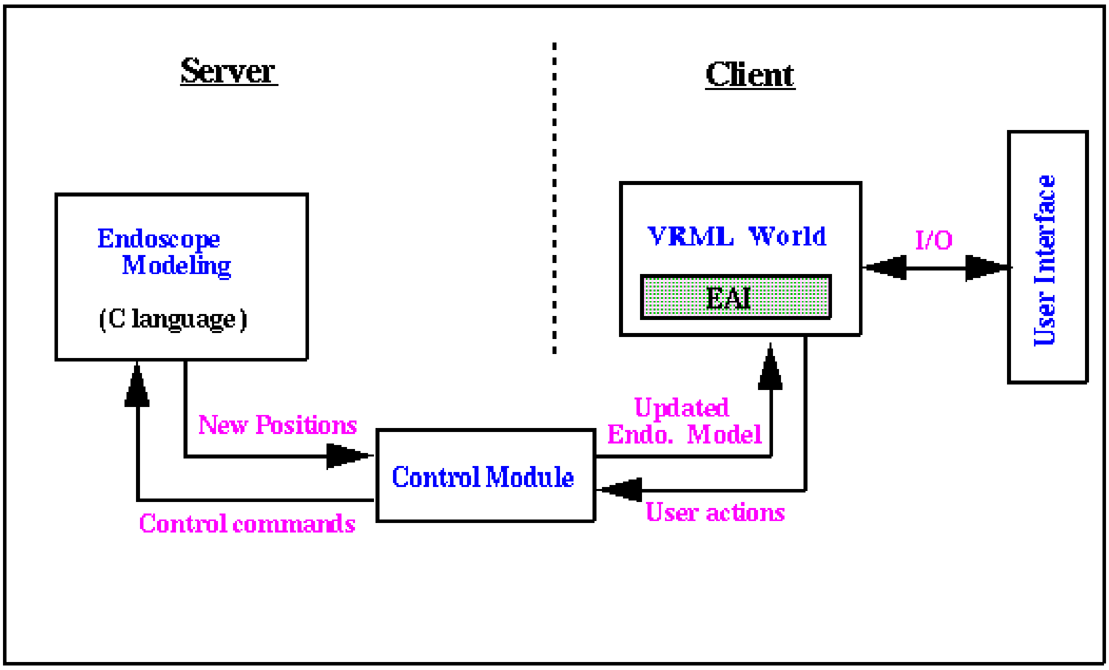

Figure (1) Data flow of the training simulator.

4.2 Implementation

4.2.1 Simulator database

The endovascular simulator database consists of two objects: the anatomical representation of the blood vessels and the endoscope representation. The geometry of the blood vessels was extracted from CT scan data obtained from St. James's

University Hospital using a 3D deformable model which adapts the resolution of the model to fit the requirements of the image data and models the branching structure of the vessel [Bulpitt, 1996]. The vessels are assumed to be rigid objects to simplify the calculations required. The triangular mesh of the vessels is represented by an indexed face set in VRML 2.0.

The endoscope geometry was also represented using an indexed face set. However, when the behaviour of the endoscope is calculated the representation is reduced to a one dimensional curve in 3D space to simplify the calculations. The endoscope curve is represented as a set of particles. Each particle has a mass, stretching, bending, damping and twisting factors which describe its behaviour when exposed to forces. Internal and external forces acting on the particles would cause the endoscope to deform- this will be further explained in the simulator processor section.

4.2.2 Simulator engine

Time events in the endovascular simulator are produced by VRML time sensors every 0.25 seconds. A virtual clock is established between the server and the client to synchronise between the incoming client commands and the server command processing. When the time stamp of the client commands fall behind the server clock, the server will ignore these commands- there is no need to process old data.

4.2.3 Simulator interface

The client interface to the simulator is provided by a standard HTML page. Two VRML worlds are embedded in this HTML page. One VRML world represents the global view of the blood vessel and the endoscope current position. Meanwhile, the second VRML world shows the internal (endoscopic) view of the vessel as if you have a camera fitted on the tip of the endoscope. In addition, the HTML page contains a Java applet running along side the VRML worlds. This applet is executing the External Authoring Interface (EAI)- expected to become standard in a later VRML specification. The EAI is used to communicate between the client and the server and to synchronise between the two VRML worlds. This will be further illustrated in the communication section.

The applet also provides user input tools. The user can apply force or torque to the free end of the endoscope via force sliders and buttons. Because of the restricted movements of the endoscope in this type of procedure (it can translate forwards/ backwards or twist around the longitudinal axis) the required tools are simple. One tool- a slider- allows the user to push/ pull the endoscope inwards/ outwards. Another slider alongside with two buttons allow them to apply clockwise/ anti-clockwise torque to twist the endoscope. These tools will be replaced by more intuitive icons in the global VRML world.

The global VRML world contains a scale which indicates the strength of the force fedback from the endoscope collision with the vessel walls. This serves as a visual substitution of the tactile feedback necessary to close the loop of interaction. In the case where the VR system of the client is sophisticated enough, this data can be transferred to a physical device held by the trainee. Meanwhile, in a low configuration

VR system, the user does not lose this feature completely, which reflects our scalability aim. Figure (2) shows the user interface of the simulator.

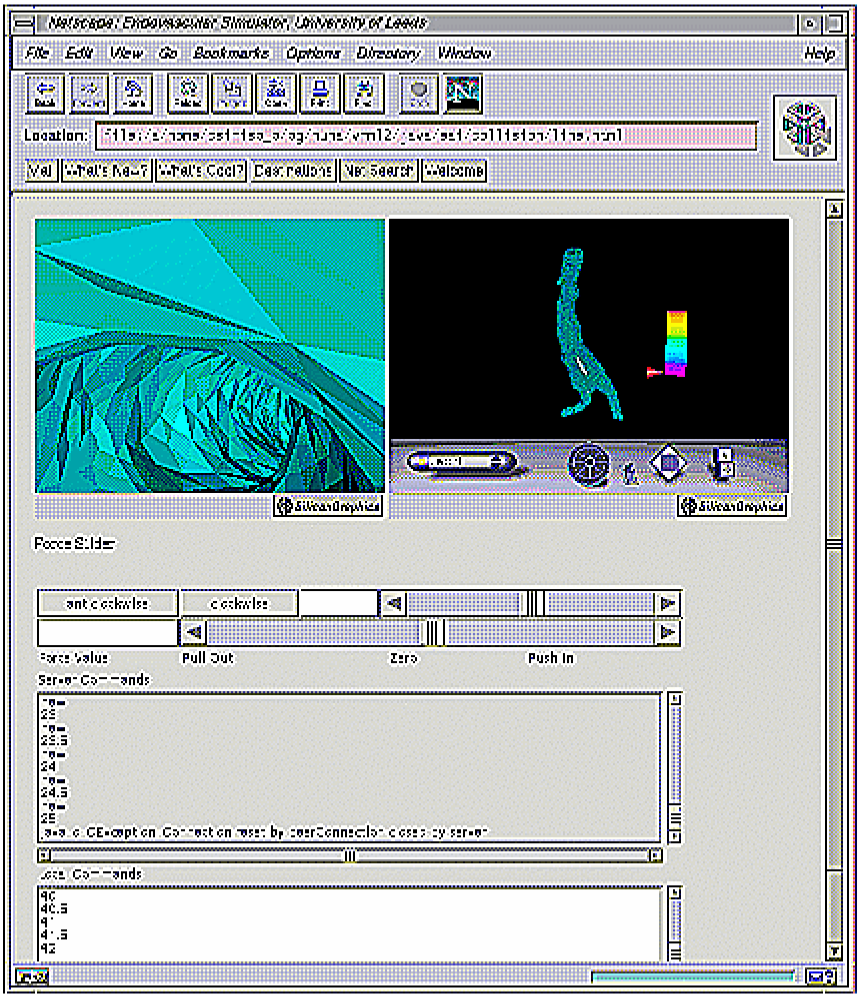

Figure (2) Client user interface.

4.2.4 Simulator processor

The task of this module is to calculate the response of the simulator objects to events coming from the engine and the interface module. Under the chosen configuration- option B- this module resides on the server side. The processor receives force and torque update commands from the interface, and accordingly will update the database of the endoscope. Meanwhile, time events received by the processor will be handled by performing the following sequence of actions:

1. Calculate internal forces due to endoscope deformation.
2. Check collision with the vessel and accordingly update the forces acting on the endoscope.
3. Calculate the new position of the endoscope according to the current forces/ torque acting on it.

When the endoscope deforms from its rest state due to external forces, internal forces are generated to counteract the effect of the external forces- i.e. attempting to return the endoscope to its rest state. These forces are calculated based on suggestions given by Terzopoulos [Terzopoulos, 1987]. Curve parameters - arc length, curvature and torsion- are calculated using the local theory of curves [Farin 1988]. Both the amount of change of these parameters from the rest state and the coefficient factors which represents the material's resistance to stretching, bending and twisting govern the strength of the internal forces.

VRML 2.0 is able to detect collision between the avatar- virtual representation of the viewer- and the other objects in the world. This limitation is unsuitable for simulation applications, where collision between objects is often of interest too. Thus we have used a collision detection tool- VCollide- to detect collision between the endoscope and the vessels [Hudson]. VCollide is a set of C libraries, which requires its models to be triangular meshes. A tree- called Oriented Bounding Box (OBB)- of the triangular model is built and is used to check model overlap for fast detection of collision. When VCollide builds an OBB tree for the endoscope, this structure remains valid while the endoscope undergoes transformations. However, because the endoscope experiences deformation rather than transformation, this structure needs to be rebuilt every time the endoscope deforms. Nevertheless, since the endoscope model was kept simple (24 triangles in the model), rebuilding the OBB tree every time step does not form an expensive overhead.

After calculating all internal and external forces/torque acting on the endoscope model, the following Lagrangian equation of motion is used to calculate the next position of the endoscope:

$$M \ d^2r / dt^2 + C \ dr/dt \ + Fi = Fe$$

where r is the global co-ordinate system which represents the endoscope- 3D positions of the particles points. M is the mass of the particle. C is a global damping factor to prevent the solution from oscillation. Fi the internal force acting on the endoscope while Fe is the external force [Terzopoulos, 1987]. The second order ordinary differential equation (ODE) is turned into first order differential equations and the Runge- Kutta integration method is used to solve the set of equations. The NAG C library is utilised to solve the differential equations.

4.2.5 Communication

Communication between the different modules of the simulator is implemented by various methods. Firstly, communication between the two VRML worlds and the Java applet is done using EAI [EAI]. This method allows access from the applet to objects or 'nodes' in the VRML scene using an event mechanism. Nodes can export their current status via 'EventOuts', which can be read by the applet; similarly the applet can send data via 'EventIns' to update the status of the VRML node. Commands from the interface and from the server all pass through the Java applet which routes them to their destination. The Java applet is also responsible for keeping both general and endoscopic views synchronised. When the server updates the position of the endoscope in the global view, the applet will also update the view point position on the second VRML world. On the other side- the server side- communication between the Java application- simulator processor- and parts of it implemented in different languages- i.e. collision detection and ODE solver- is done using native methods provided by the Java language [Java].

Communication between the client and the server is done via sockets which utilises the Java networking capabilities. Because this part of communication is done over the Internet, extra care should be taken of the amount information sent back and forth to reduce the expected delay of network communication. Communication from the client to the server is kept to a minimum of sending time commands or force update commands associated with the required values- e.g. force value. On the other hand, communication from the server to the client consists of the new positions of the endoscope particles. By representing the endoscope as a one dimensional curve for behavioural calculation, we have reduced this communication to a small set of 3D points. Nevertheless, it is crucial that the simulator client and server sides are kept synchronised. Hence, the time stamp mechanism described earlier is essential.

5. Conclusion and Future Work

Our endovascular prototype runs at a rate of 4 frames per second. The simulator processor takes about 10 msec. to perform one calculation step.

We have adopted VRML 2.0 and Java in our implementation to achieve the portability required in a web-based application. However, because different web browsers do not fully implement the VRML standard, complete portability is not currently achieved. Nevertheless, when the VRML standard is finalised and implementations gain extended functionality, users on different platforms with different browsers should be able to fully utilise the application.

During the implementation stage, we have faced some limitations due to VRML and Java features. For example, VRML can only detect collision between the view point and the objects of the world. Even in this case, no collision information- e.g. collision position- is available to be used for dynamics calculations. Another drawback of VRML 2.0 is the lack of real time sensor- VRML time sensor is browser dependent and is not guaranteed to produce real time events- which hinders its usefulness for real time applications. The current standard of VRML does not support multi- view worlds; to overcome this limitation we have utilised the EAI to communicate between two separate virtual worlds which represent different views of the same world. Furthermore, because of the security features of Java applets, native methods can not be used to implement pieces of code in other languages that might be more efficient than Java. The only way around this restriction is by keeping the native methods on the server, which might not be the efficient solution to real time applications.

The next stage of our research will investigate intuitive interface tools suitable for our endovascular application. Iconic tools that are part of the virtual environment will be developed to enable the user to push/pull and twist the virtual endoscope.

Then, we will reach the final stage of developing a simulator, where subjective evaluation of the endovascular simulator will be carried out using trainee questionnaires.

ACKNOWLEDGMENT

The authors would like to thank David Kessel from St. James's University Hospital, Andy Bulpitt, Tim David, Peter Jimack and Stuart Lovegrove for their co-operation.

REFERENCES

[Bulpitt, 1996] Bulpitt, A. and Efford, N. An Efficient 3D Deformable Model with a Self- Optimising Mesh. Image and Vision Computing. 14:573-580, 1996.

[CieMed] CieMed: DaVinci: Medical Simulators.
<URL: http:// beast.cbmv.jhu.edu:8000/ projects/ daVinci/daVinci.shtml>

[Dumay, 1995] Dumay, A. and Jense, G. Endoscopic Surgery Simulation in a Virtual Environment. Computers in Biology and Medicine, 25(2): 139-48, 1995.

[EAI] External Authoring Interface, Working Group.
<URL: http://www.vrml.org/WorkingGroups/vrml-eai/ >

[Edwards, 1996] Edwards, J. And Hand, C. MaPS: Movement and Planning Support for Navigation in an Immersive VRML Browser, 1996.
<URL: http://www.cms.dmu.ac.uk/Research/CSTR/1996/>

[Farin, 1988] Farin, G. Curves and Surfaces for Computer Aided Geometric Design. Academic Press, Inc. , 1988.

[Hon, 1996] Hon, D. Medical Reality and Virtual Reality. Proceedings of Medicine Meets Virtual Reality IV. IOS Press, San Diego Jan, 1996.

[Hudson] Hudson, T. and Lin, M. and et. al. V-COLLIDE: Accelerated Collision Detection for VRML.
<URL: http://www.cs.unc.edu/~dm/collision.html>

[Java] The Java Tutorial: Object-Oriented Programming for the Internet.
<URL: http://java.sun.com/docs/books/tutorial/index.html>

[Meglan, 1996] Meglan, D. Making Surgical Simulation Real. Computer Graphics, Nov:37-39 1996.
<URL: http://www.ht.com/ htweb/Test.html >

[Murray, 1997] Murray, H. Peloton: a VRML-Based Bicycling Simulator. 1997.
<URL: http://www.lucent.com/press/0897/970806.bla.html>

[Pegden, 1995] Pegden, C. Shannon, R. and Sadowski,R. Introduction to Simulation Using SIMAN. McGraw-Hill, Inc. New York, 1995.

[Schumann, 1997] Schumann, M. Web-based Simulation Environment, 1997.
<URL: http://www.cs.uni-magdeburg.de/~maschuma/GraphIt/GraphIt.html>

[Simjava] Simjava. University of Edinburgh.
<URL: http://www.dcs.ed.ac.uk/home/hase/simjava/simjava-1.0/>

[Terzopoulos, 1987] Terzopoulos, D. and Platt, J. et. al.
Elastically Deformable Models. Computer Graphics, 21(4):205-214, 1987.

[Walsum, 1997] Walsum, T. and Zuiderveld K. et. al. CT-based Simulation of Fluoroscopy and DSA for Endovascular Surgery Training. Proceedings CVRMed- MRCAS'97
1205: 273-282, 1997.