Collaborative Virtual Training Using Force Feedback Devices

Maria Andréia Formico Rodrigues[1], Ricardo Régis Cavalcante Chaves[1], Wendel Bezerra Silva[2]

[1]*Mestrado em Informática Aplicada*
Centro de Ciências Tecnológicas
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares 1321, sala J30
60811-905 Fortaleza-CE-Brasil
mafr@unifor.br, rchaves@edu.unifor.br

[2]*Bacharelado em Informática*
Centro de Ciências Tecnológicas
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares 1321
60811-905 Fortaleza-CE-Brasil
wendel@edu.unifor.br

Abstract

Force feedback plays an important role in collaborative virtual reality environments, mainly for programmers of haptic visualization tools. Whereas a great deal of work has gone into graphical displays over the past years, little has changed on the input side. One of the problems that has slowed down development in this area is the difficulty of integrating the visualization of a scene, the interaction of the user with the scene, the feeling for the user to be immersed inside the scene, and finally, the input devices. In this paper, we describe the architecture we have designed, implemented and tested for a collaborative virtual training using force feedback devices. In particular, it provides device independence and easy extensibility through a compartmentalized and multilayered model. We also present examples of how force feedback joysticks can be integrated into training exercises using our prototype.

1. Introduction

A collaborative virtual environment can be defined as a single virtual reality space shared by multiple participants connected from different hosts. Most collaborative existing systems however restrict the communication between the participants to text messages or audio communication [1]. The natural means of human communication are richer than this. During collaborative training, for example, other effects of coordinated visual and touch feedback play also an important role and create a more realistic experience to the users. More specifically, during training sessions, the users are expected to perform some tasks under the supervision of a trainer during navigating and interacting realistically with the virtual environment. In this case, realism not only includes believable appearance and simulation of the virtual world, but also implies the visual embodiment of the users and the means of interaction with the world and feeling various attributes of it using the senses. Actually, collaborative virtual training is an area that puts special demands on input [2] and so does on output, when using force feedback devices.

We believe that collaborative virtual training using force feedback devices may benefit from being able to manipulate work models, feel the form and contact of collision, weight, surface friction, texture, and softness or hardness of objects remotely. Motivated by this, we are particularly interested in the development of a collaborative virtual training system in which users using any type of force feedback device can not only manipulate and explore a single virtual reality environment, but can also make realistic touch contact with it and with the other users and objects. To address it, we present some related work in the area, and generically describe force feedback devices with emphasis on a commercial model used in our work, the Microsoft SideWinder Force Feedback II [3] (section 2 and 3, respectively). Then, a collaborative architecture for training is proposed, implemented and tested (section 4). In particular, it provides device independence and easy extensibility through a compartmentalized and multi-layered design. In our implementation, according to a communication protocol over the network, a trainer (master) can control a session attended by many trainees (slaves). The trainees are expected to perform some tasks under the trainer supervision, during navigating and interacting via force feedback with the virtual reality environment. A trainer can also temporarily hand the control over to one of the trainees, either by the trainer's own initiative or upon request by the trainee. As collaboration is achieved, there is no need for the trainer and trainee to be present at the same location.

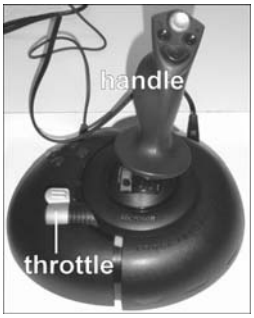

Figure 1 The force feedback joystick used as the input-output device. Photographed by one of the authors.

The prototype was evaluated using three force feedback joysticks working collaboratively during two training sessions (section 5). More specifically, one session was carried out in playback mode while the other one was realized in real-time, including geometric collision effects. Finally, conclusions and future directions for collaborative virtual training using force feedback devices are given (section 6).

2. Related work

We are particularly interested in related work on collaborative virtual environments and on using force feedback devices for interacting with tri-dimensional virtual spaces during training.

Most collaborative virtual reality systems consist of basic components such as a virtual reality space stored in a computer, a device or interface, a communication protocol, and the user. These components are integrated using multiple program layers. In particular, there are some platforms and applications that have been developed for robust distributed virtual worlds. Examples are MASSIVE [10], EQUIP [11], DIVE [12], OpenMASK [13], among others. MASSIVE has support for data consistency, and world structuring. It adopts a distributed database model, in which all changes to items in the database are represented by explicit events that are themselves visible to the system [10]. It can also support a certain number of mutually aware users using real-time audio. EQUIP is a dynamically extensible open-source framework for integrating C++/Java applications with a variety of interfaces and devices, ranging from wireless portable devices through to fully immersive and large systems [11]. DIVE is a collaborative virtual environment based on communication protocols that already incorporate facilities for sharing states in a heterogeneous network environment [12]. OpenMASK is an open-source middleware for the development and execution of modular applications in the fields of animation, simulation, and virtual reality [13]. Collaboration between distant users within virtual environments is possible with OpenMASK in which several users can share simultaneous interactions with the same interactive object.

A major problem with these generic and large systems is that they are generally not open-source (MASSIVE, DIVE), nor well documented (MASSIVE, EQUIP, DIVE). Hence, they are difficult to be re-used or extended to other scenarios. Recently, for portability reasons, some developers have launched a Java version of their code (EQUIP) which is still under testing. Other systems, although reasonable documented, only run under Linux/Unix operational system (OpenMASK). Finally, most systems remain mainly limited to sharing text-based data and audio, without including force feedback effects.

Recent enhancements to virtual environments allowing users to touch, feel and manipulate the simulated objects using mechanical devices (haptic or force feedback devices) that mediate communication between them and the computer have been mainly proposed in the Haptics area [2,4,5,6]. Force feedback devices, beyond having the abilities of a standard input device such as a mouse or an ordinary joystick, are also output devices [7]. This characteristic enables them to track a user's physical manipulation (input) and provide realistic touch sensations coordinated with on-screen events (output). Each force feedback device has its own strengths and weakness, just as each application has its own unique demands. Devices incorporating force feedback are all net force displays, in that they mediate the virtual touch on an object by a tool, the tools being the handle of an input-output device [8]. A number of studies have shown that adding haptic force feedback improves single users' performance during training [14,15,16,17,18].

3. Force feedback devices

We classify the force feedback devices according to the number of degrees of freedom (DOFs) that they offer force feedback. The most common devices are the joysticks that have two DOFs and the force feedback applied to both. These DOFs enable the joystick to restrict movements, exert forces or to apply waveforms to simulate different conditions. Professional systems often have three DOFs, sometimes six, and force feedback in at least three of them. These devices can simulate volumes, and not only objects in the plane to which we are constrained in the joystick. As a user manipulates the handle of a

force feedback device, encoder output is transmitted to an interface controller at very high rates [5]. The information is then processed to determine the position of the end effector that is sent to the host computer running a supporting software application. If the supporting software determines that a reaction force is required, the host computer sends feedback forces to the device. Actuators (motors within the device) apply these forces based on mathematical models that simulate the desired sensations. For instance, when simulating the feel of a rigid wall, the motors apply a force that resists the penetration. The farther the user penetrates the wall, the harder the motors push back, to force the device back to the wall surface. The end result is a sensation that feels like a physical encounter with an obstacle.

The basic idea of a force feedback joystick is to move the stick in conjunction with onscreen action. The Microsoft Sidewinder Force Feedback II joystick (see Fig. 1) used in this work is one of several force feedback devices currently on the market. It is a low cost device developed only in the early 00´s. It has a USB port and an on-board 16-bit processor running at 25 MHz. This processor handles all the force effects. There are three force effects that can be represented by this input-output device [3]. First, there are time-based effects such as jolts and vibrations. These are not really related to the orientation of the joystick handle, but instead depend on the temporal profile of the force. Second, there are space-based effects like springs, dampers, and walls. These present a changing force depending on the orientation of the joystick handle and how fast it is moving. Finally, there are invariant effects, constant forces like wind or gravity. Beyond these effects, the SideWinder Force Feedback II joystick supports a number of effects that may be combined to generate new ones. These effects vary from simple raw forces in an arbitrary direction, to complex force-waves in spatially located walls. The co-processor takes care of all the control, decides if the joystick is inside or outside the wall, and applies corresponding forces. Up to four walls are supported concurrently [9]. As with sensible movements, we can consider many different properties including DOFs supported, range, speed, accuracy and stability. We can also consider how the physical form of the application affords and constrains some basic movements such as translate sideways ($\uparrow$x), raise and lower vertically ($\uparrow$y), push and pull forwards and backwards ($\uparrow$z), tilt forwards and backwards (αx), rotate on vertical axis (αy), and tilt sideways (αz).

A virtual environment contains information about the magnitude and direction of forces to be applied to the user, usually depending on the position and velocity of a cursor in the environment. Every time the user moves the handle of the joystick, the position of the cursor changes, allowing for dynamic interactions with the virtual reality environment. The information about the position, as well as the force to be displayed, usually has an update of at least 500Hz for smooth haptic display [3]. A major issue occurring in this case, is the update frequency of the computers which is generally more than an order of magnitude lower than the update frequency of the force feedback device [4].

The strength of the joystick force is called magnitude and it varies according to a percentage value. It is measured in units that run from zero indicating no force, to 10,000 indicating the maximum force for the device [3]. A negative value indicates force in the opposite direction. Magnitudes are linear, so a force of 6,000 is twice as great as one of 3,000. All effects have a duration that is measured in microseconds. Periodic effects have a period, or the duration of one cycle, also measured in microseconds. The phase of a periodic effect is the point along the wave where playback begins. A ramp force has beginning and ending magnitudes. The basic magnitude of a periodic effect is the force at the peak of the wave. Finally, a force can be constrained within a set of range over time by using “envelopes”. They are used to specify attack and fade values to modify the beginning and ending magnitude of the effect. These values have a duration which is used to define the time that the magnitude takes to reach or fall away from the sustain value.

In the next section we briefly describe the design and the implementation details of the collaborative virtual training prototype using force feedback devices we have developed.

4. Components of the architecture

In our implementation, Java is the core technology of our collaborative virtual training architecture as well as the library for creating and manipulating tri-dimensional geometry in a platform independent way using Java3D, which is designed to provide support for applications requiring higher levels of performance and interaction [19]. The proposed architecture is composed of four components, as shown in Fig. 2: a Device Interface (that enables the Java Virtual Machine, JVM, to access the force feedback device), a Virtual Reality Environment (that also handles collision detection and response), a Device Handler (that is responsible for mapping the movements performed by the user in the virtual environment and

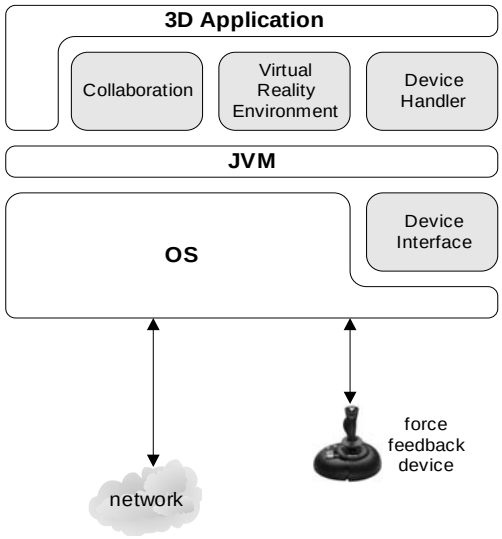

Figure 2 The compartmentalized and multi-layered design of the collaborative virtual training architecture using force feedback devices.

for mapping the feedback effects to the Java3D), and a Collaboration layer (that consists of a communication protocol responsible for data sharing and control).

There are some interesting *Application Programming Interfaces* (APIs) for interacting with force feedback devices [20,21,22]. One of the APIs investigated as a possible choice for a component of our collaborative training architecture was the Immersion API [20]. Unfortunately, despite its robustness, it is only available commercially. Other APIs investigated were Linux APIs [21]. However, few of them are available for interacting with force feedback devices and even fewer are compatible with the Microsoft SideWinder Force Feedback II joystick model. Further, these Linux APIs are unluckily ill-documented. Finally, a well-documented API that particularly allows Windows based systems to run and display rich applications in multimedia elements is the DirectX [22]. Aware of the main limitations of these APIs, the DirectX (version 9.0) was the one chosen to interface to the SideWinder Force Feedback II joystick (see the Device Interface component in Fig. 2). In our application, the DirectX API provides force feedback support specifically using the *DirectInput* interface [23]. Generally, custom device drivers for every input device involve native code. In particular, under Win32 it is necessary to implement a layer to the *DirectInput* API to allow the use of a device. In Fig. 2, the *Java Native Interface* (JNI) is used to interface to the Device Interface component (written in C++) that in turn calls DirectX methods.

Advanced input-output devices require advanced programming. The difficult issue is how to implement a program capable of setting up and handling an input-output device. Further, there are a large number of parameters that need to be set correctly. Most importantly, there are two main code segments required to develop a force feedback graphical application: the routine(s) to create force feedback effects and the routine(s) to play them back either from code control or triggered by a user hardware event (e.g., when the user presses a joystick button).

Usually, an input-output device consists of the lowest level interface with the data source. In the Device Handler component of Fig. 2, a specific element can be represented by a sensor on the force feedback device. More specifically, a device processes the raw input and fills in the sensor information. In particular, an input device can provide information to the sensors in one of three fashions: blocking, non blocking, and demand driven [24]. We have particularly chosen the demand driven implementation. It guarantees that data is always available but is only presented to the runtime environment when it is specifically requested by the application. Comparing to the other mentioned approaches, the demand driven implementation causes the least load on a runtime environment.

Our designed architecture supports an input-output device that takes the input from the joystick hardware and supplies information on demand to the runtime environment. With *DirectInput*, the force feedback device can react to an application in which the user defines effects such as jolts, vibration, or resistance when an object collides with an obstacle, or a button or trigger is squeezed. In *DirectInput* terms, a particular instance of movement or resistance over a period of time is called an effect. *DirectInput* defines a number of standard categories of effects, called forces. Some of these forces are described as: constant force (a steady force exerted in a single direction), ramp force (a force which increases or decreases in magnitude), periodic effect (a force that pulsates according to a defined wave pattern), and saw-tooth-up/saw-tooth-down (a waveform which drops/rises vertically after reaching a maximum positive/negative force) [3,23].

The Collaboration component (see Fig. 2) is responsible for all exchanges of information among users. It consists of a communication protocol over the network (see Fig. 3), the directory server that corresponds to an entity (master) that holds information about all participants in a training session, and the communication controller.

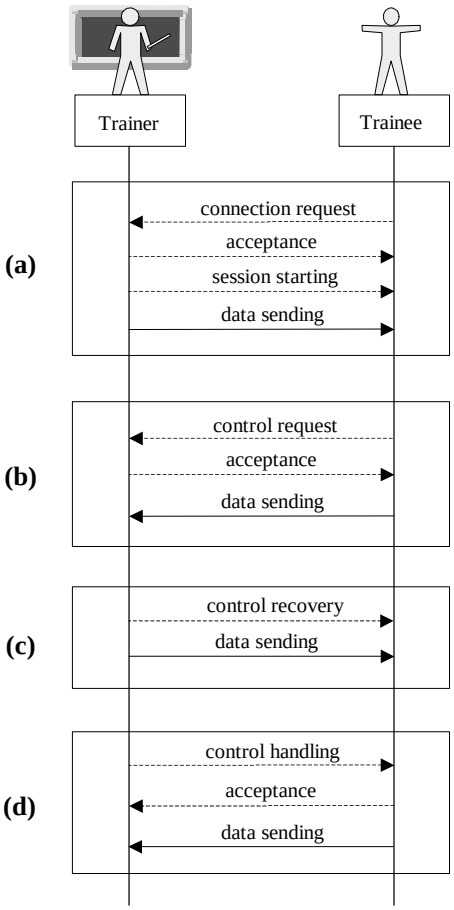

Figure 3 The communication protocol over the network using a TCP stream connection for the *command* channel (dashed lines) and UDP datagrams for the *data* channel (solid lines). In (a), the trainer creates a session and accepts the entrance of a number of trainees (clients). The trainee may request the training control to the trainer, as displayed in (b). The trainer may accept this request or not. In (c), the trainer can take the control back from a trainee at any time. The trainer can also temporarily hand the control over to one of the trainees by the trainer's own initiative, as shown in (d).

The native platform communication library is loaded into the Java environment using the JNI through the Device Interface component. Users´ actions are sent to all participants of a collaborative session through the communication protocol module. We have specified in our implementation two types of information passed between the application and the force server (master). In particular, commands affecting system state (starting, initiating local force and force feedback computation) should be delivered intact and not lost. By contrast, position reports and updates to intermediate representation parameters are sent frequently, so a lost packet can be ignored since a new one will arrive shortly. Currently, we use two channels between the client and master, i.e., the *command* and *data* channels. More specifically, a TCP stream connection for the *command* channel (reliable, high overhead) and UDP datagrams (unreliable, low overhead) for the *data* channel, as shown in (a), (b), (c) and (d) of Fig. 3. In (a), the trainer creates a session and accepts the entrance of a number of trainees (clients). The trainee may request the training control to the trainer, as displayed in window (b). The trainer may accept this request or not. In (c), the trainer can take the control back from a trainee at any time. The trainer can also temporarily hand the control over to one of the trainees by the trainer´s own initiative, as displayed in window (d). Our system prototype provides an asynchronous continuous report, in which the master sends position reports at regular intervals, using the *data* channel, rather than upon request. As discussed by Mark [25], this mode avoids the wait for a round-trip network message, usually required by standard requests. The application can poll these continuous reports or block them. In particular, we currently use only one channel for UDP datagrams (for the force feedback and joystick positioning updates). However, the architecture proposed in this work can be easily extended to support several UDP channels, for instance, for collaborative audio transmission as well. In our implementation, the actions and feedback interactions among users are communicated to other participants to have the impression of being involved in a training exercise. The status of the training exercise is transmitted into the Collaboration layer, as shown in Fig. 2. The trainer has the role of the master (see Fig. 3). The other participants get this status at the beginning of their sessions and initialize the training scenario with these settings. For example, to explore the virtual reality environment (a maze we have generated automatically using Java3D), the master can use the handle of the joystick to change his positioning (through rotations and translations) and interact with the environment through force feedback. The orientation of all the other participants in their respective scenarios is set into the system in real time and so is the feedback. Using our prototype system, we are also able to record a training session for later playback through a synchronization layer.

During our collaborative virtual training, collision effects between users and maze walls need to be detected and taken into account through touching or interpenetrating interactions (see the Virtual Reality Environment component of Fig. 2). Besides being

Figure 4. Three participants (one trainer and two trainees) during a collaborative virtual training using the SideWinder Force Feedback II joysticks. The trainer is using the handle of the joystick as a flight simulator controller to navigate on and feel through force feedback a virtual maze. When the trainer finds obstacles with the maze walls, the collision effects felt by the master are transmitted collaboratively to the trainees through force feedback. Photographed by one of the authors.

detected, and contact area determined, collisions have to be handled for collision response that induces instantaneous change in the state of components through direct correction of position and speed. These interaction forces need to be calculated at high rates to satisfy the control requirements of haptic interface hardware [16]. We have implemented a traditional approach in order to simplify computational costs involved during contact. First, we use a bounding sphere algorithm to determine whether a point is near to a surface maze wall. Then, we calculate the exact collision detection point. In particular, the different sensors on the joystick are used to detect the distance to the closest maze wall in the direction of motion. If any sensor detects an object closer than *d* (a pre-defined critical distance), the motion is stopped. Otherwise, *d* is used to calculate the velocity to be set which represents the force response of the system to the collision.

Standard sensors are used to drive the user´s view position in our implementation. In particular, Java3D has a set of standard sensor inputs that may be used to automate some of the control during a collaborative training session. Basically, it is used to provide a socket to place any given sensor and allow it to control the interactions with the scene graph. Our prototype uses a standard sensor that is usually the most interesting because it allows to use head move along type systems to automatically track where the user is looking. Wherever the joystick moves and orientates, the viewpoint is moved with it. There are various ways to react to sensor input. As our application is using a force feedback joystick it may be desirable to read sensor data every single frame and react to it. Other times, it may be more convenient to the application to react to sensor input by creating behaviours that only launch when the sensor enters a particular bounding region. In our prototype, the former type of reaction happens during the whole training session, while the latter happens every time that a geometric obstacle is found. In either case, the system requires the use of behaviours to prepare the application code to read information from the sensors and react to it by applying this information to the scene graph as well as to the feedback response of the joystick.

All user interactions with the graphical application layer are performed using the joystick and its buttons. All the feedback is done by the haptic device, which can be made to move and react to events. The user is free to explore the structure as he can feel the walls being simulated by force feedback in the joystick. When an exit is found, this is indicated by an oscillation. All the structures are simulated in the bi-dimensional plane that the joystick handle moves in. The absolute position within the movement range of the handle is used as the desired position in the virtual structure. We have mapped three DOFs of the joystick (with force feedback applied to two of them) in a very intuitive way that mimics a flight control system. The throttle button (with one DOF for translations) and the handle of the joystick (with two DOFs for rotations), as shown in Fig. 1, were mapped to perform roll and pitch movements, respectively. More specifically, the handle of the joystick is used to map movements such as tilt forwards and backwards (αx) as well as sideways (αz), and the throttle button is used to map movements such as push and pull forwards and backwards ($\uparrow z$). In our implementation, the velocity is

a parameter that can be also controlled and modified by the throttle button. It is measured in units that run from zero indicating no velocity, to 65,000 indicating the maximum velocity for the joystick [3]. Similarly to the force magnitude, the velocity varies according to a percentage value. The frame of reference for the movement analysis during the collaborative virtual training is that of the device itself. All these mappings are implemented in the Device Handler component (see Fig. 2).

5. Collaborative virtual training

Collaborative virtual training can be used to construct a virtual world where users can share the environment in which they preside as well as to enhance the way they "feel" the data or objects when performing training exercises.

In our implementation, we designed a collaborative system that allows users to navigate a maze, with their respective joysticks providing feedback. In the graphical scenario, routes are determined following a specific trajectory chosen by the master user. Using the force feedback joysticks and the sense of touch, users are able to feel the effects of phenomena (such as viscous damping, stiffness, and inertia) at the same time the master is feeling these effects. Indeed, feeling the dynamics improves user's understanding and adds an element of a great interest to the training exercise.

Our collaborative training session can also be performed through pre-recorded spaces. During the playback, the frame rate is kept at constant rates. A trainer (the master) has control over frame rate through the force feedback joystick. In addition to speed control, as the trainer takes the handle of the joystick and moves it from side to side, the position of the handle is sensed by all the other users. Based upon the position and velocity of the handle, various amounts of force are reflected back to the users.

A realistic demonstration is built with three participants handling their respective force feedback joysticks simultaneously, as shown in Fig. 4: one trainer (master) and two trainees (slaves). Basically, the training goal is to navigate on and feel through force feedback a virtual reality maze. During the training session, collision effects between users and maze walls are taken into account, making the collaborative virtual training appears as real as possible. The haptic properties modelled are texture, size, weight and stiffness. To begin the task, the master guides the participants to explore the collaborative scenario. The users can feel the surface of objects/walls in the common environment in a collaborative fashion using the force feedback joystick.

6. Conclusion

A collaborative architecture for the control of force feedback devices has been proposed and tested in a virtual training scenario. In particular, it provides device independence and easy extensibility through a compartmentalized and multi-layered design. Force feedback adds a lot of value to any graphical application and is certainly worth the effort to implement it. The combined effects of coordinated visual, and touch feedback create a realistic experience.

We believe that collaborative training will be a valuable concept for both the developers of haptic devices and the end-users of such devices. In our training scenario, the low cost commercial force feedback joysticks serve as haptic interfaces and provide the users with real-time feeling of the virtual reality environment interactions. In spite of this, collision detection is often the bottleneck of simulation applications in terms of calculation time, directly related to the scene complexity. In particular, it is a critical point for virtual environment applications where real-time performance is required. The higher the complexity of the computer graphics in a scene, the lower is the perceived force feedback response of the joystick.

Performance and subjective measures are currently being carried out to quantify the scalability and the role of force feedback in our prototype system. The preliminary results show that the force feedback joystick model used intuitively indicates the user the applied force during training sessions. However, there are important joystick hardware limitations mostly due to limited maximum force capability. As joysticks continue to evolve, it is expected that manufacturers will take force feedback technology to whole new levels. Indeed, force feedback controller technology may lead to significant changes in industrial machinery, games and medical care. The benefits and the number of possible collaborative applications using haptic devices are endless. For instance, surgical simulations and medical training, development of virtual reality environments for people with special needs (e.g., to assist blind people), and virtual art exhibitions, are some of the areas where feedback devices are making an appearance. In the short term, our hope is to develop a generic and robust collaborative virtual environment using haptic devices for training. Libraries of objects can be then created

and used to provide the component parts for a variety of virtual environments that may be shared, simulated, felt, analyzed and visualized by the virtual world of trainee and instructor using force feedback devices ubiquitous as computer keyboards are today.

References

[1] C. Joslin, I.S. Pandzic and N.M. Thalmann, "Trends in Networked Collaborative Virtual Environments", *Computer Communication Journal*, Vol. 26, No. 5, pp. 430-437, 2003.

[2] R. Baecker, J. Grudin, W. Buxton and S. Greenberg, "Touch, Gesture & Marking", *Human-Computer Interaction: Toward the Year 2000*, pp. 469-482, 1995.

[3] The Microsoft SideWinder Force Feedback II joystick. Available at http://www.microsoft.com/ hardware/sidewinder/Joysticks.asp. Last visited on 12th May 2004.

[4] L. Fluckiger and L. Nguyen, "A Generic Force-Server for Haptic Devices", *SPIE Telemanipulator and Telepresence Technologies VII*, Boston, 2000.

[5] J.J. Berkley, "Haptic Devices", White Paper by Mimic Technologies Inc., pp. 1-4, May 2003.

[6] E-L. Sallnas and S. Zhai, "Collaboration Meets Fitts' Law: Passing Virtual Objects With and Without Haptic Force Feedback", In *Proc. of INTERACT 2003, IFIP Conference on HCI*, pp. 97-104, 2003.

[7] P.J.Kovach,"*Inside Direct3D*", Microsoft Press, 2000.

[8] A. J. Johansson and J. Linde, "Using Simple Force Feedback Mechanisms as Haptic Visualization Tools", *16th IEEE Instrumentation and Measurement Technology Conference*, Venice, Italy, 1999.

[9] B. Bargen, P. Donelly. "*Inside DirectX*", Microsoft Press, 1998, ISBN 1-57231-696-9.

[10] C. Greenhalgh and S. Benford, "MASSIVE: A Collaborative Virtual Environment for Teleconferencing", *ACM Transactions on Computer-Human Interaction*, Vol. 2, No. 3, pp. 239-261, ACM Press Publisher, New York, USA, September 1995.

[11] C. Greenhalgh, S. Izadi, T. Rodden, and S. Benford, "The EQUIP Platform: Bringing Together Physical and Virtual Worlds", Technical Report, 2001. Available at http://machen.mrl.nott.ac.uk/home.html. Last visited on 12th May 2004.

[12] C. Carlsson and O. Hagsand, "DIVE - A Platform for Multi-User Virtual Environments''. *Computers & Graphics* Vol. 17, No. 6, pp. 663-669, 1993.

[13] D. Margery, B. Arnaldi, A. Chauffaut, S. Donikian and T. Duval. "OpenMASK: Multi-Threaded Animation and Simulation Kernel: a General Introduction", *VRIC 2002 Proceedings*, 2002.

[14] C. Basdogan, C. Ho, M.A. Srinivasan, and M. Slater, "An Experimental Study on the Role of Touch in Shared Virtual Enviroments", *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 4, pp. 443-460, 2000.

[15] Microsoft DirectX-DirectInput MSDN documentation. Available at http://msdn.microsoft.com/. Last visited on 12th May 2004.

[16] G.C. Burdea, "Haptic Feedback for Virtual Reality", In *Proc. of the Virtual Reality and Prototype Workshop*, Laval, France, pp. 87-96, June 1999.

[17] F. Vahora, B. Temkin, T.M. Krummel, and P.J. Gorman, "Development o Real-Time Virtual Reality Haptic Applications: Real-Time Issues", In *Proc. of the 12th IEEE Symposium on Computer-Based Medical Systems*, IEEE Ed., pp. 290-295, 1999.

[18] G. Burdea, "*Force and Touch Feedback for Virtual Reality*", John Wiley & Sons, New York, USA, 1996.

[19] G. Rowe, "*Computer Graphics With Java*, Palgrave Macmillan, 2001.

[20] Immersion TouchSense Technology. Available at http://www.immersion.com. Last visited on 11th May 2004.

[21] F. Brachere, Microsoft Force Feedback 2 Driver for Linux Project. Available at http://madfab.free. fr/ff/. Last visited on 11th May 2004.

[22] Microsoft DirectX API. Available at http://msdn.microsoft.com/archive/enus/directx9_c/direc tx/input/using/forcefeedback/. Last visited on 9th May 2004.

[23] Microsoft DirectInput Force Feedback MSDN Documentation. Available at http://msdn. microsoft.com/archive/enus/directx9_c/directx/input/usi ng/forcefeedback/effecttypes.asp. Last visited on 12th May 2004.

[24] J. Couch, "Input Devices". Available at http://www.j3d.org/tutorials/raw_j3d/chapter9/input_dev ices.html. Last visited on 13 th, May 2004.

[25] W.R. Mark, S.C. Randolph, M. Finch, J.M.Verth, and R.M. Taylor II, "Adding Force Feedback to Graphics Systems: Issues and Solutions", In *Computer Graphics Proceedings*, New Orleans, Louisiana, pp. 447-452, ACM SIGGRAPH, August 1996.