Collaborative Virtual Environments for Industrial Training and e-Commerce

JAUVANE C.OLIVEIRA[1,2], XIAOJUN SHEN[1] AND NICOLAS D.GEORGANAS[1]

[1] Distributed & Collaborative Environments Research Lab.
School of Information Technology and Engineering
University of Ottawa
161 Louis Pasteur Priv.
Ottawa, ON K1N 6N5
CANADA
+1 (613) 562-5800 x 6248

[2] COMCIDIS Research Group
Department of Computer Science
National Laboratory for Scientific Computation
Av. Getúlio Vargas, 333
Petrópolis, RJ 25651-070
BRAZIL
+55 (24) 2233-6022

jauvane@acm.org shen@discover.uottawa.ca georganas@discover.uottawa.ca

Abstract: Collaborative Virtual Environment (CVE) concepts have been used in many systems in the past few years. Applications of such technology range from military combat simulations to various civilian commercial applications. This paper presents CVE as a medium for Industrial Training and Electronic Commerce.

Key-Words: Collaborative Virtual Environments, Virtual Reality, Collaboration, Java3D, VRML, Multimedia, Industrial Training, E-Commerce.

1 Introduction

Over the past few years, a number of interactive virtual reality (VR) systems have been developed. A Collaborative Virtual Environment (CVE) is a special case of a VR system where the emphasis is more on "collaboration among users" rather than on simulation. CVEs are used for applications such as collaborative design, training, telepresence, and tele-robotics.

We have exploited CVEs for Industrial Tele-Training aiming at reducing expenses and saving time while training remote users/engineers to operate some equipment. We also have been exploiting CVEs for Electronic Commerce aiming at providing the user with a much enjoyable experience while shopping online. In both cases the users, represented by avatars, can join a Virtual World and then manipulate and interact with the objects as in the real world. For training purposes the users are expected to perform some tasks under supervision of a trainer (or unattended) while the e-Commerce client would be able to see a model of the products he/she wishes to buy. If questions need to be answered an Intelligent Agent may pop-up and assist the user with detailed information and walk through features of the virtual objects. One can extend the Virtual World to be similar to a real shopping mall, which would facilitate the adaptation of users not used to browser-based interfaces.

Motivated by these advantages, we have designed and implemented prototypes in those two areas, namely an Industrial Teletraining prototype for ATM switching equipment, as well as a Virtual Shopping Mall. Section 2 focuses on the Industrial Training prototype, while section 3 focuses on the e-Commerce one.

2 Industrial Training

From an industry perspective, CVEs can be an attractive solution for reducing training expenses [1]. Instead of users working with physical objects, the latter are represented by virtual objects that can then be placed in a virtual environment accessible to many users. The users, represented by avatars, can then manipulate and interact with the objects as in the real world, gaining valuable experience and training before using the real equipment

Motivated by these advantages, we have designed and implemented an industrial teletraining prototype. At the request of our industrial sponsors, the prototype has been created for training operators of ATM switching equipment; however it can be modified for other types of training. Such a prototype can be fully controlled in a wireless fashion where speech commands and user gestures are used as input, allowing for a more comfortable interaction among the human participants.

2.1 Prototype

Our prototype is a multiuser teletraining application, which allows users, represented by avatars, to learn how to repair a faulty ATM switch. The avatars repair the switch in steps, which precisely reflect the steps necessary to perform the same actions in the real world. The prototype consists of two general modules: user interface, and network communication. The user interface itself consists of a graphical interface (GUI), a 3D interface (VR), and media interfaces (speech recognition, voice streaming, head tracking). Figure 1 shows the user interface of the prototype.

The upper right area of the interface, which takes the largest part, is the *3D environment*. On the left and below the 3D environment are the *controls* used by the trainer and trainees to interact with objects and navigate in the environment. At the top left is the *head-tracking facility* which will be discussed in more detail later in sections 2.2.6 and 2.3. Below the head-tracking window there is window for playing training video clips (e.g., how to “ground” a user before touching an electronic part) Below that there is a *utility panel* that is used for different purposes as will be discussed later. There is also a *chat space* where users can exchange textual messages. In order to avoid navigation problems with inexperienced users, it is possible to view the world from a set of predefined camera views as shown in Figure 2a.

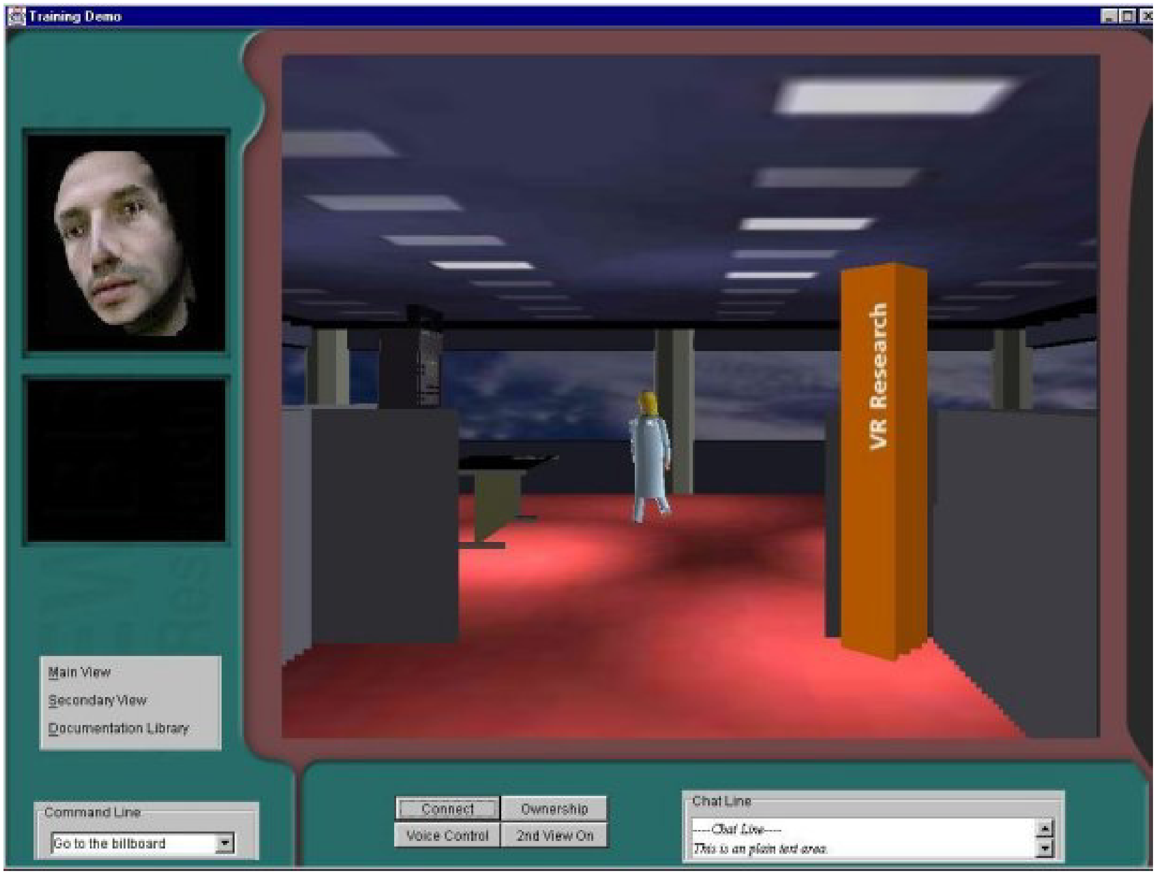
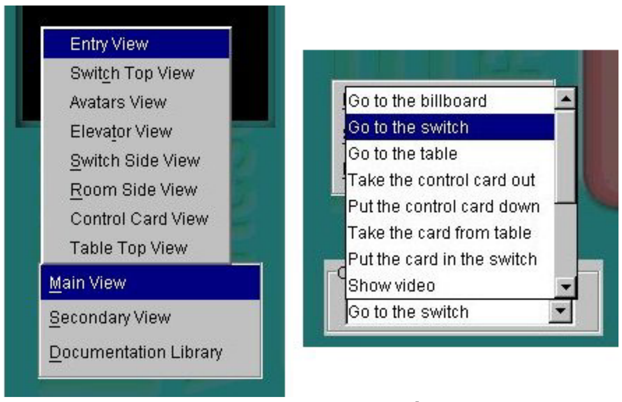


Figure 1. Training Application’s Interface

A user is able to approach and verify the operation of the switch and its cards, remove a faulty card, put it on the repair table, and replace it by installing a new card into the switch. Other parties will be able to watch that user's avatar performing such actions. All of the above actions can be performed by directly navigating in the scene and manipulating objects with the mouse or by selecting the action in a menu as shown in Figure 2b. Figure 3 shows a sequence of screenshots illustrating the execution of the previously mentioned steps performed by a trainer and watched by a trainee.



a b

Figure 2. Camera View/Action Choice Menu

As can be seen from figure 3, a trainee can adjust his/her viewpoint to whichever angle/position that is more suitable in order to watch the trainer's actions. These adjustments can be done quickly by choosing the appropriate views from the camera view menu. When the trainer selects a new view of the World, the trainees get the same view as well. That was implemented based on feedback from users and trainers, which is aimed at enhancing the trainee's experience by viewing the actions through the best angle suggested by the trainer. A trainee may change the view if he/she so desires.

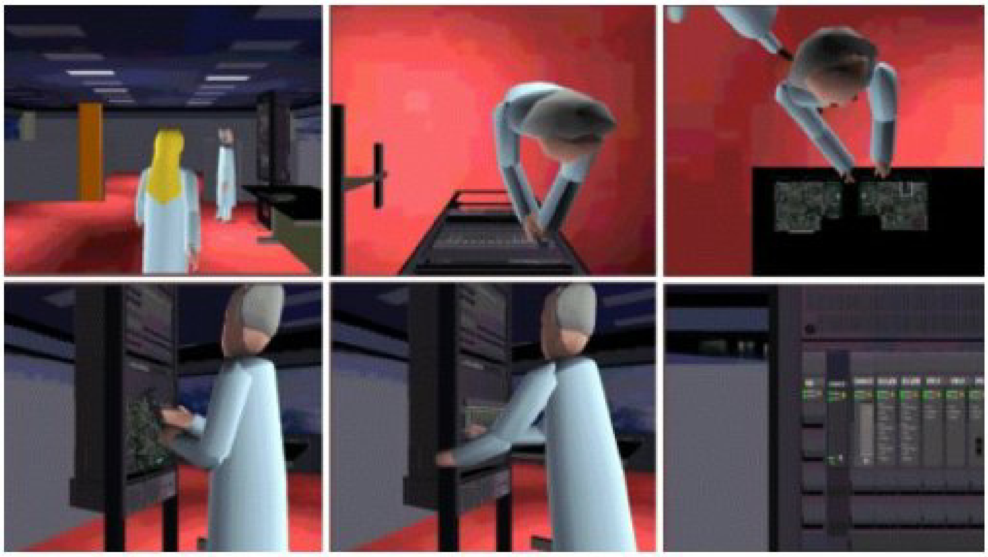


Figure 3. Replacing a Faulty Card.

In addition to receiving help from the trainer, a user can also view video segments showing the correct procedure for performing certain actions. The utility panel is used to display the video clips, as shown in Figure 4b. If chosen by the trainer, the video will be displayed in every participant's screen.

Another use for the utility panel is the *secondary-view* feature, where a user can simultaneously watch the current actions from an alternative point of view as shown in Figure 4a.

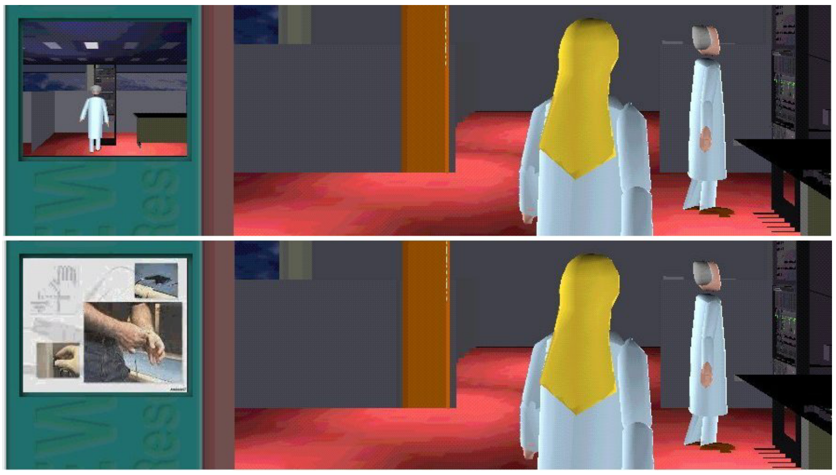


Figure 4. Utility Panel functionality:
a) secondary view (top) b) video clips (bottom)

In addition to the above explained interfaces, the prototype offers voice recognition technology, whereby the user simply enters commands by talking into the computer's microphone. In this case, the user may simply give verbal pre-defined commands, such as for example "Go to the table", for the avatar to perform, or may change views by saying "Table Top View", and so on.

2.2 Architecture

Rather than developing the system from scratch, we decided to make use of a wide spectrum of available technologies, reuse software packages, and concentrate our efforts in integrating them. Hence, the system brings together a large number of components and technologies as shown in Figure 5, namely for 2D graphics and interface design, 3D rendering, multi-user communications, voice recognition, audio and video streaming, and head tracking. In this section we will briefly describe the prototype architecture and the role of each component. Such components are detailed in Figure 7.

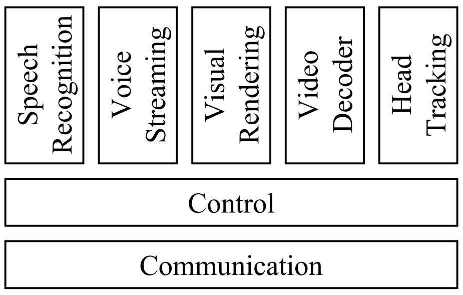


Figure 5. System Architecture.

2.2.2 Communication

The communication layer is responsible for all exchange of information amongst users. There are three components in this layer: directory server, control communication, and media communication layers.

The *directory server* is an entity which holds information about the participants in a session; the directory server is also responsible for checking on the status of every participant through alive bits.

The *control communication layer* is responsible for all communications with the exception of audio data. Packets have the format shown in Figure 6, where the field *command* indicates what kind of information lies within the packet.

Command	Int Parameter 1	Int Parameter 2	Str Parameter 1	Str Parameter 2	Str Parameter 3

Figure 6. Control Packet Format.

The directory server has been written in ANSI C and the client side of the control communication in C++, resulting in a Dynamically-Linked Library (DLL) to be loaded at run time. There are IPv4 and IPv6 implementations available so that, depending on their capabilities, a group of users can choose whichever version they find appropriate by selecting the corresponding DLLs and directory server. The IPv6 implementation will eventually make use of quality of service parameters, multicast groups and flow labels when the users would benefit from better behaviour of the system.

The communication layer decouples all networking issues from the upper layers. The native-platform communication DLL is loaded into the Java environment using the Java Native Interface (JNI). Incoming messages are sent to the Java core via callback of Java methods from the native code.

2.2.3 Rendering and Interface

For the GUI part, we used JavaSwing, which is a set of high-level graphics APIs for more advanced and sophisticated graphical components. Java technology was chosen due to its easy utilization of high-level APIs. When designing the 3D interface, we examined both Java3D and various VRML browsers. Other works have shown that Java3D is faster, easier to control and manipulate, and more interactive than the interface offered by VRML browsers [2]. Java3D also eliminates certain complexities that are inherent to using VRML along with its Java External Authoring Interface, Web browser and VRML browser. On the other hand Java3D has no specific file format and a developer must convert 3D descriptions of other famous formats to Java3D code.

Our prototype makes use of VRML objects designed in PowerAnimator. In order to make such objects treatable under Java3D a comprehensive optimization is performed in the original VRML objects aiming at simplifying the geometry (reducing redundant polygons). The VRML behavior of the objects is converted to Java3D code using VRML to Java3D conversion tools. A view of a world created in this manner is then presented in a Canvas3D. Hot spots are defined in the world in order to allow the user to interact with them in a point-and-click fashion. User actions are sent to the communication layer, which in turn informs the other parties involved in or affected by such actions, creating thus a collaborative environment. Figure 7 shows the various components of the system along with the communication paths among them.

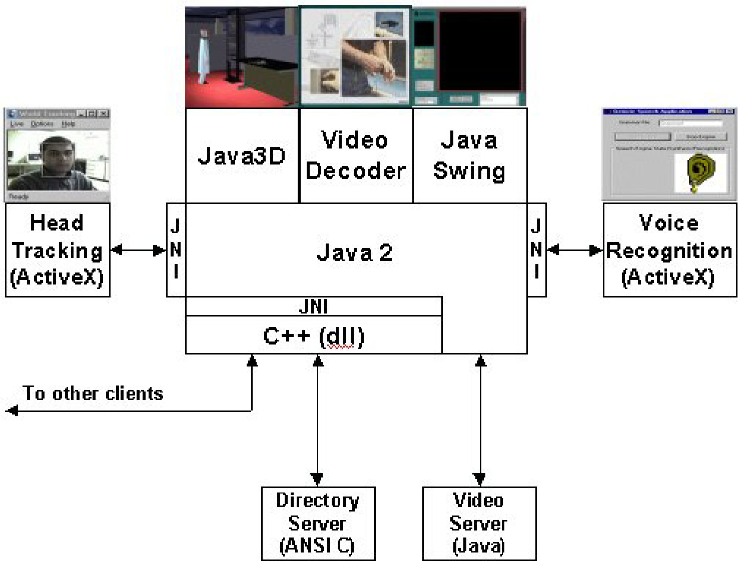


Figure 7. Prototype Modules/Connection

2.2.4 Speech Recognition

We use the Microsoft Speech API (SAPI) to provide a speaker independent recognition of pre-defined commands described by a SAPI grammar. This way commands such as "go to the table", "pick up the card", are recognized by the SAPI module. An example of SAPI code is shown below:

```
[<Animation>]
<Animation>=go to the "go to the" <Select3>
<Select3>=billboard "billboard"
<Select3>=switch "switch"
<Select3>=table "table"
```

In the above example, all the "go to the X" commands are programmed into the SAPI engine, in which X is one of the three options, namely, “billboard”, “switch”, or “table”. These commands will invoke the appropriated actions.

The speech recognition module is implemented using ActiveX components and its integration to the prototype is made via another native DLL loaded by JNI. The speech recognition module works independently from the other modules, by sending recognized commands into the prototype via a pre-defined local socket.

2.2.5 Audio Conferencing

The audio capturing module allows participants to enter into an audio-conferencing session. The module is based on the Microsoft NetMeeting SDK. Due to Soundcard limitations, this module will compete with the voice recognition module described above. Text chat facilities are also optionally provided and can be used in cases where audio is not supported.

2.2.6 Video Player

The Video Decoder module is an H.263 Video Decoder developed entirely in Java which is able to receive and decode streamed video from a video server [13]. The video is presented in the utility panel.

2.2.7 Head Tracking

The head-tracking module captures the user’s head motion wirelessly by video processing techniques using a simple camera installed on the user's computer. Head movements are sent to the prototype and are used to control the corresponding avatar’s head. The head-tracking module is implemented using ActiveX components which generate a series of rotation parameters which are sent to the prototype via a JNI connection through yet another DLL. More details are disclosed in section 2.3.

2.2.8 Control

The control layer is implicitly included in the other layers and is composed of a set of rules which ensures that all components work together. The coordination of the DLLs, which enable the exchange of data between the Java core of the prototype and the native components, comprises the Control Layer.

2.3 Video Processing

Model-based video coding (MBVC) has recently emerged as a very low bit rate video compression method suitable for Collaborative Virtual Environment (CVE) applications [9]. The MBVC increases coding efficiency by using knowledge about the scene content and describing the real world geometry of 3D model objects. The principle of this compression is to generate a parametric model of the image seen at the emission end and to transmit only the characteristic parameters that show how the model changes in time. These differential parameters are then used to animate the model of the image recovered at the reception end.

The first step in a full automatic MBVC system is the *face detection* allowing the identification and location of the face in first image frames. The next step is *motion estimation* encompassing global 3D-motion recovery, local motion estimation, expression and emotion analysis, etc. The problem is technologically difficult, as 3D motion parameters have to be extracted from a sequence of 2D images of the performer’s head-and-shoulders.

In our system [14], we use a *3D tracking method* for the real-time measurement of six head motion parameters, namely 3D position and orientation, and the focal length of the camera. This method uses a 3D wireframe head model, a 2D feature-based matching algorithm, and an Extended Kalman Filter (EKF) estimator. Our global motion tracking system is meant to work in a realistic CVE without makeup on speaker's face, with uncalibrated camera, unknown lighting conditions and background.

The EKF converts the 2D feature position measurements, using a perspective camera model into 3D estimates of the position and orientation of the head [3, 4, 5]. The EKF recursive approach captures both the cause-effect and the dynamic nature of the tracking, offering also a probabilistic framework for uncertainty representation.

The EKF procedure is applied to nonlinear systems and consists of two stages: time updates (or prediction) and measurement updates (or correction). At each iteration, the filter provides an optimal estimate of the current state using the current input measurement, and produces an estimate of the future state using the underlying state model. The values, which we want to smoothen and predict independently, are the tracker state parameters.

The EKF state and measurement equations can be expressed as [14]:

$$s(k+1) = As(k) + \xi(k) \qquad (1)$$

$$m(k) = Hs(k) + \eta(k) \qquad (2)$$

where s is the state vector, m is the measurement vector, A is the state transition matrix, H is the Jacobian that relates state to measurement, and $\xi(k)$ and $\eta(k)$ are error terms modeled as Gaussian white noise.

The observations are the 2D feature coordinates *(u,v)*, which are concatenated into a measurement vector $m(k)$ at each time step. The observation vector is the back-projection of the s state vector containing the relative 3D camera-scene motion, and the camera internal geometry, namely the focal length. In our case the state vector is $s(translation, rotation, velocity, focal\_length)$ that contains the relative 3D camera-object translation, rotation and their velocities, and camera focal length.

The EKF requires a physical dynamic model of the motion and a measurement model relating image feature locations to motion parameters. Additionally, a representation of the object (user's head) is required.

The dynamic model is a discrete-time Newtonian physical model of a rigid body motion, moving with constant velocity. The measurement model relates the state vector s to the 2D-image location (u_k, v_k) of each image feature point, using a perspective projection model.

We employ a three-parameter incremental rotation $(\omega_x, \omega_y, \omega_z)$, similar to that used in [4] and [6] to estimate inter-frame rotation. The incremental rotation computed at each frame step is combined into a global quaternion vector (q_0, q_1, q_2, q_3) used in the *EKF* linearization process and rotation of the 3D-model [7].

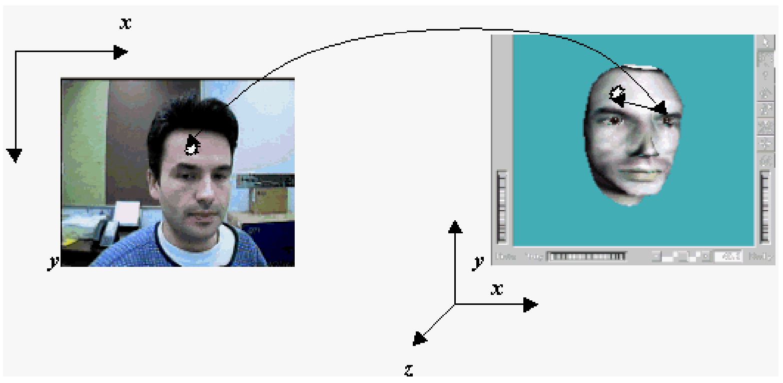


Figure 8. Hotspot in Video and Correspondent 3D Head

2.3.1 EKF Initialization

The 3D-model provides the initial structure parameters of the Kalman filter. Each 2D-feature point (u_i, v_i) corresponds to a structure point $p_i(X_i, Y_i, Z_i)$. As shown in Figure 8, these (u_i, v_i) points are obtained by intersecting the 2D image plane with a ray rooted in the camera's center of projection COP and aiming to the 3D structure point on the head model. Figure 9 shows the EFK Initialization.

Step 1. **The user positions his/her face** at the center of the screen, and adjust the matching of the live image and the projected mesh , so that the projected mesh covers the entire facial region.

Step 2. **Left mouse click** on every rigid feature point of interest on the live image. An automatic program function takes care to properly align the selected live feature point to a vertex of the projected mesh.

Step 3. **Right mouse click** anywhere on the active Windows "live" image triggers the tracking process (booting EKF module).

Figure 9. The EKF "Boot" Algorithm

The typical point identification problem of the 3D pose recovery from 2D images is solved in our case by identifying corresponding points in both the 2D live image of the subject and the 3D model of the subject's head. In order to aid the point identification process, we are using an augmented reality technique by projecting in the 2D live image the 3D mesh used to model the head. A multiple "point identification" procedure using this augmented reality technique is summarized in Fig. 8. At this development stage it is still up to the user to arrange the scale matching between the live face image and the projected mesh. Figure 10 shows the user's head moving and its avatar matching the 3D model.

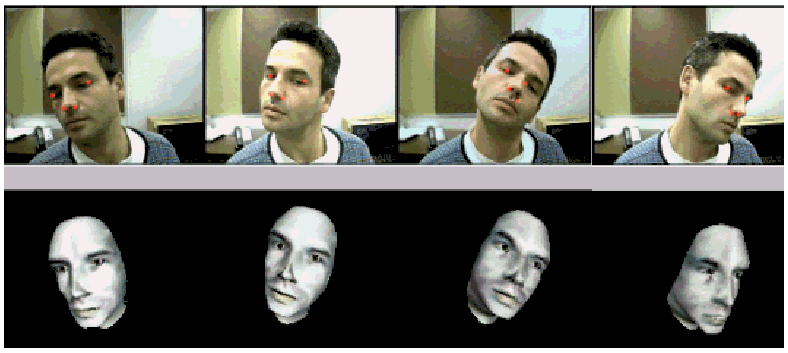


Figure 10. Tracking the head motion

2.3.2 EKF Update

At each iteration, the EKF computes an estimate of the rigid 3D motion that must probably correspond to the motion of the 2D live image. We employ the Kanade-Lucas-Tomasi (KLT) [8] 2D-gradient feature tracking method, which robustly performs the tracking reinforced by the EFK estimation output. An estimate of motion and camera focal length is found at each step. After the 3D-motion and focal length are recovered, a perspective transformation will project feature points back onto the image to determine an estimated position of the 2D feature trackers. At the next frame in the sequence a 2D tracking is performed starting at this 2D estimated position. The current matching coordinates of tracked features are fed back into the Kalman filter as the observation vector, and the loop continues. The feedback from EFK is used to update the 3D-model pose parameters, i.e. provides the 3D head tracking information.

The recovered 3D position and orientation are propagated to the *Head Modeling* block of the CVE system, which renders a new posture of the 3D-model as illustrated in Fig. 9.

2.4 Immersive Environment

A later enhancement to this prototype was the possibility to have the same environment in a more immersive fashion. Figure 11 shows the immersive version of this prototype projected in a wall as well as in a computer' screen. The immersive version of the prototype benefits from alternative input methods such as voice recognition and head-tracking previously introduced. This Prototype shows the potential that CVE has for Industrial training.

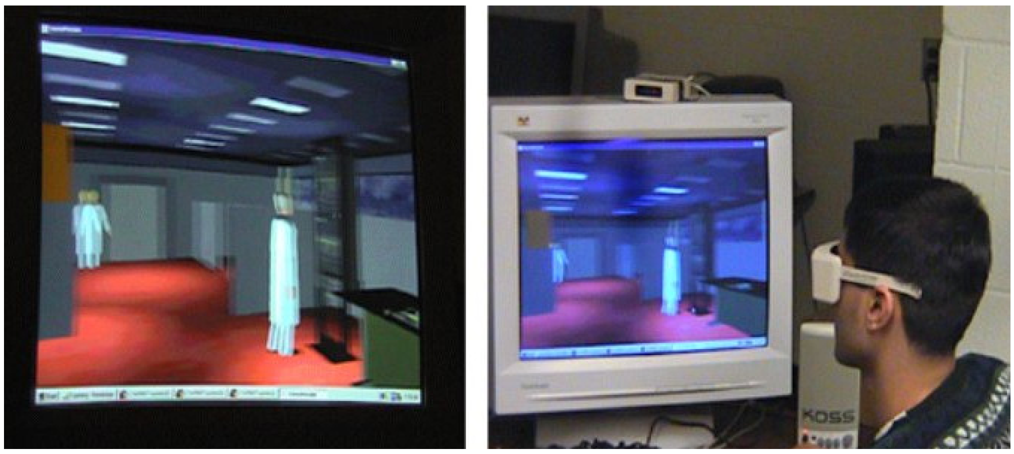


Figure 11. Immersive version of Prototype

3 e-Commerce

Existing electronic commerce applications only provide the user with a relatively simple browser-based interface to access the available products. Buyers, however, are not provided with the same shopping experience, as they would have in an actual store or shopping mall. With the creation of a virtual shopping mall, simulations of most of the actual shopping environments and user interactions can be achieved. The virtual mall brings together the services and inventories of various vendors. Users can either navigate around from one vendor to the other, adding items into virtual shopping carts, or perform intelligent searches through "user search agents". The user may find items of interest in the normal navigation process or use the search agent without having to navigate through the maze; he/she can then negotiate with sales agents to bargain for the best price and make a secure transaction. The virtual mall prototype also allows the user to communicate with an "intelligent assistant" using simple voice commands. This assistant interacts with the shopper using voice synthesis and helps him/her use the interface to navigate efficiently in the mall.

Real-time interactions among entities in the virtual environment are implemented over the Run Time Infrastructure of High Level Architecture (RTI/HLA), an OMG and IEEE standard for distributed simulations. This High Level Architecture (HLA) standard is a general architecture for simulation reuse and interoperability developed by the US Department of Defense.

The HLA standard [11, 15] promotes reuse of simulations and their components. It attempts to specify the general structure of the interfaces between simulations without making specific demands on the implementation of each simulation.

3.1. Virtual Mall Overview

This Virtual Mall project focuses on e-commerce and the infrastructure that is needed to implement an Internet-based version of this e-commerce application. The main aspects of technology that pertain to e-commerce were addressed as follows:

- Using user interface design, virtual reality, and intelligent agents to make electronic commerce applications more attractive and usable,
- Ensuring electronic payments are secure -- essential to the success of electronic commerce,
- Using data warehousing, mining, and management to add value to an application, or to handle the application's data flow,
- Designing architectures for various applications and solutions,
- Measuring quality of service, given available resources.

This section only discusses the CVE-based user interface and intelligent agents design.

Figure 12 provides the overview of the Virtual Mall project. A user uses his/her web browser, logs on to a given URL and follows proper authentication/access control. Registration is required if it is a new user. The user needs to give his/her personal information such as name, birthday, or credit card number. The information will be stored in the User Profile Database. The user then may choose the browsing mode that he/she prefers – search engine or virtual mall. Search engine plays a major role in the traditional electronic shopping on the Internet. The user can check all sorts of information he/she wants, evaluate the products or services he needs, compare prices and makes a purchase right away on a "flat" web page.

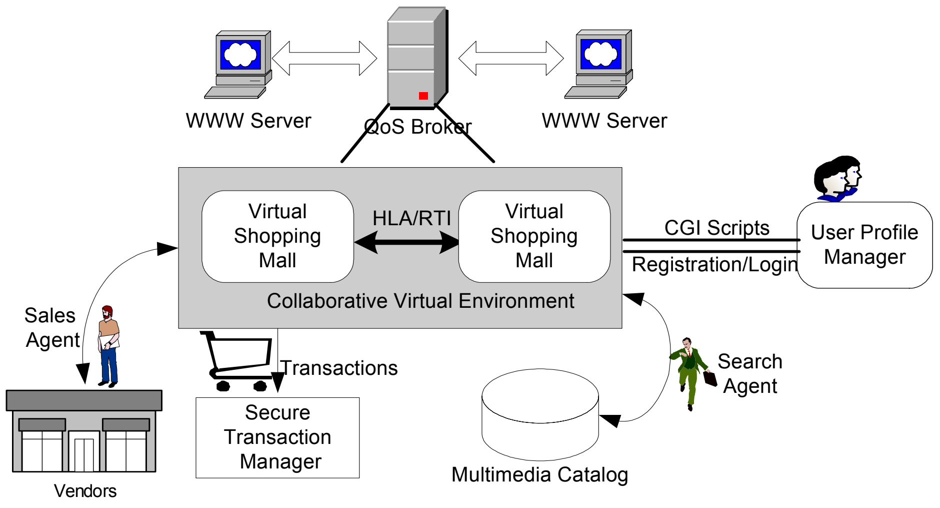


Figure 12. Overview of Virtual Mall Project

Virtual Mall is a CVE user interface through which the user can retrieve the virtual world of interest realized in VRML (Virtual Reality Modeling Language). He/she then can navigate the virtual world, where both the users and agents are represented on the screen by 3D animation avatars. Human to human interaction is possible in the distributed virtual environment, such as communication between users via non-verbal means, like gestures.

In terms of Quality of Service (QoS), an architecture that introduces a brokerage function between clients and servers has been defined. QoS brokers continuously monitor the performance of affiliated servers and assign them to clients according to a defined server selection policy that optimizes the capacity of the system based on server performance characteristics and user requirements.

There is a number of data management issues that need to be addressed in electronic commerce applications. These include, among others, the development of virtual catalogs, intelligent query access of these catalogs, and support for the management of transactions between customers and vendors. The multimedia catalog enables users to access multiple, distributed and potentially heterogeneous catalogs in a uniform and transparent manner. The user queries the database through CGI scripts. The user search agent is responsible for generating the CGI scripts, parsing the responses from database, which is in CGI format as well, and showing the results on a query window. The user may pick up his interest items into the virtual shopping cart and negotiate with vendor sales agent to bargain for the best price. After that, the Secure Transaction Manager helps the user to make the deal.

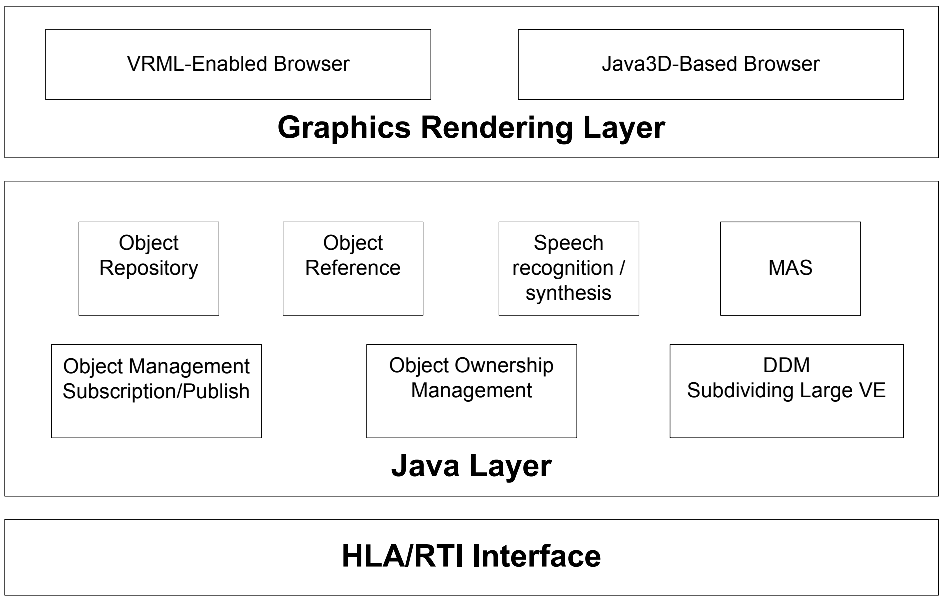


Figure 13. System Architecture

3.1.1 System Architecture

The CVE is a shared 3D virtual world over HLA/RTI on the Internet where users from homes, offices, can explore, interact and work. Figure 13 depicts the basic three-layer system architecture.

The top layer is the graphics rendering layer, where there are two kinds of browser modes that the user can choose: VRML-enabled web browser, i.e. Netscape, or Java3D-based browser. Even thought a Java3D-based browser is a pure Java solution, the scene description of virtual worlds is written in VRML (Figure 14).

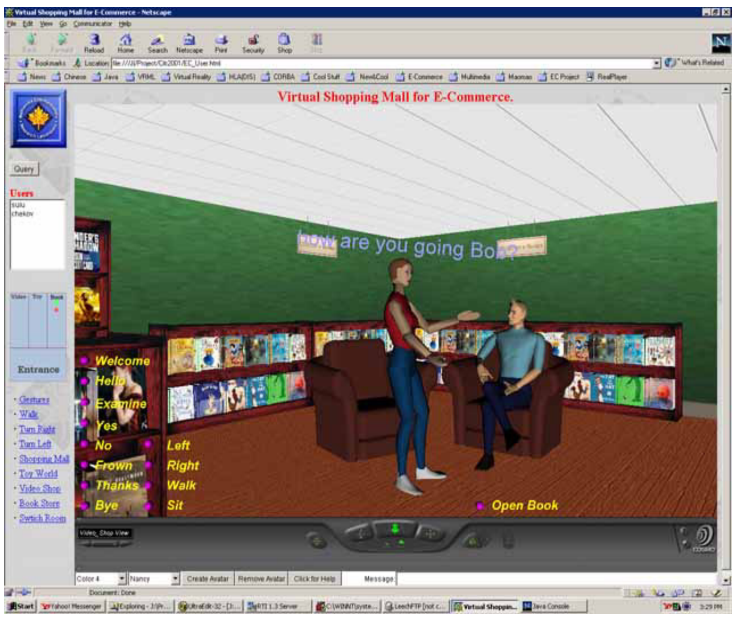


Figure 14. Browser-Based User Interface

The Java layer plays a pivotal role in the creation of CVEs. It not only controls the virtual scenes (VRML scenes) but also communicates with the RTI through the Java Native Interface (JNI). The Java layer also implements the service that RTI provides, such as RTI Object Declaration, Object Ownership Management and Data Distribution Management (DDM), which will be addressed in detail in the following sections.

3.1.2 Object Management in CVE

The virtual world consists of various types of entities, each of which has a visible body containing its state. In particular, it contains its geometry and physical characteristics. One entity often needs information about other entities: it is affected by them, and may need to interact with them. At first sight it appears that there is a fundamental distinction among *active* entities (such as avatars), *passive* entities (such as the objects that need to be co-manipulated) and *inactive* entities (like the background objects in the virtual worlds).

Different entities may have different behaviors, for example, the way they move, fly or walk. Different objects are used to embody a certain entity in the virtual environment at each of the three layers of the application – VRML, Java and RTI. For example, at the VRML layer, a VRML PROTOTYPE node defines the avatar's geometry and the animation of its actions/gestures, the latter of which are implemented by VRML sensors, events and scripts. At the Java level, a Java object consists of attributes like position, rotation and gesture to describe the avatar inside the CVE. Simulated entities in the RTI context are represented by RTI objects containing attributes and state definitions.

HLA Declare Management (DM) provides the services to federates to declare their intention to generate and consume information [12]. Object Management (OM) deals with the registration, modification, and deletion of object instances and the sending and receiving of interactions. The basic activities of DM and OM services for object class operations are shown in Figure 15.

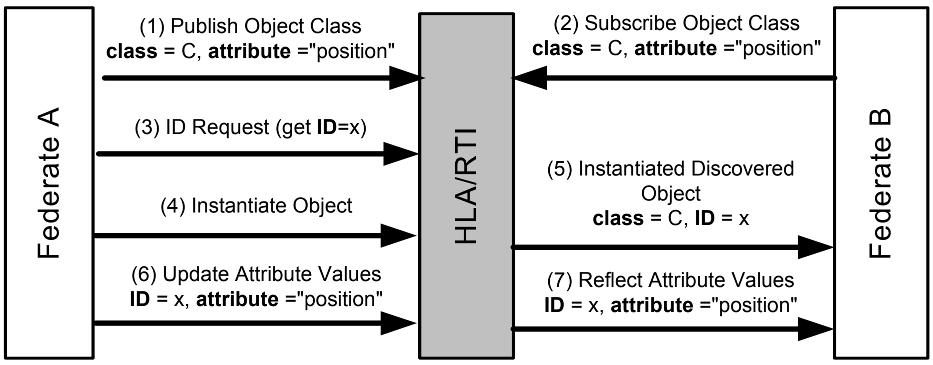

Federate A
HLA/RTI
Federate B
(1) Publish Object Class
class = C, **attribute** ="position"
(2) Subscribe Object Class
class = C, **attribute** ="position"
(3) ID Request (get **ID**=x)
(4) Instantiate Object
(5) Instantiated Discovered Object
class = C, **ID** = x
(6) Update Attribute Values
ID = x, **attribute** ="position"
(7) Reflect Attribute Values
ID = x, **attribute** ="position"

Figure 15. RTI/HLA Object Declaration

The execution mechanism of VRML in a standalone browser supports sensors, events and scripts, which allow the animation of a local scene graph. To support our goal of shared interactive scenes, we allow scripts to communicate events to the scene graphs managed by other browsers via RTI. Our approach provides a simple mechanism to share user's (avatar) interaction, which offers methods for non-verbal clues between collaborators, within all browsers in the 3-D world. The model is a replicated script mode with each browser downloading the same script and executing it locally. Typically these scripts would be associated with objects that are in the initial downloaded VRML file.

In Figure 16, we can see the message flow as a result of a user selection. A user selection (1) causes a local script to run (2). This in turn invokes the callback event by External Authoring Interface (EAI) in the Java applet, which then retrieves the event (3) and asks the RTI to send attribute update through JNI (4). RTI passes the event and invokes the remote federate ambassador to reflect updated attribute. The latter one is "RTI initiated" service (5). The Java applet retrieves the message to its browser (6), which then converts the message to an event that causes execution of the local script (7).

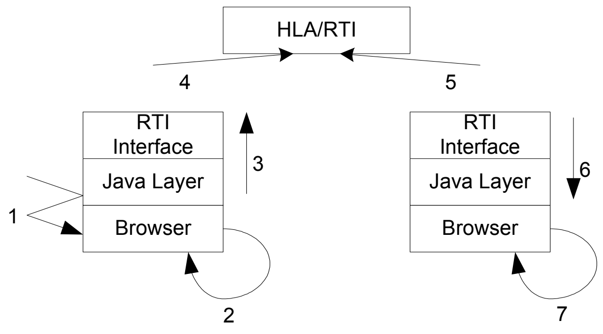


Figure 16. Sharing User Interactions

3.1.3 Ownership Transfer Mechanism

Multi-user CVE worlds is subject to conflicts. Conflicts, for example, may arise if one user tries to open a window while another user is trying to close it. These conflicts must be avoided or resolved. One method to determine the attributes of objects that may be modified by certain users is to assign temporary ownership to objects. Manipulation of objects may include the change of the object's coordinate system and the change in the scene graph: users may grasp objects and take them to a different world. Operations like this are essential for applications like virtual shopping.

One of the attributes of an object in the virtual worlds is its ownership, which indicates which user (process) is allowed to access or manipulate the object. The manipulation is performed through modifying some of the attributes of an owned object (such as position and rotation). In the RTI implementation, a given object instance has one single owner at any given time. However, attribute ownership may change during the execution. This mechanism allows two or more users to handle the same object through the transfer of ownership within the CVE.

The RTI allows federates to share the responsibility for updating and deleting object instances with very few restrictions. It is possible for an object instance to be wholly owned by a single federate. In this case, the owning federate has responsibility for updating all attributes associated with the object and for deleting the object instance. It is possible for two or more federates to share *update responsibility* for a single object instance. When update responsibility for an object is shared, each of the participating federates has the responsibility for a mutually exclusive set of object attributes. Only one federate can have update responsibility for an individual attribute of an individual object at any given time. Also, only one federate has the *privilege to delete* an object instance at any given time.

The ownership management methods provide a facility for exchanging attribute ownership among federates in a federation execution using a "*push*" and/or a "*pull*" model. A federate can try to give away responsibility for one or more attributes of an object instance – or *push* ownership. Alternatively a federate can try to acquire responsibility for one or more attributes of an object instance – or *pull* ownership. The *push* model cannot thrust ownership onto an unwitting federate. Similarly, the *pull* model cannot usurp ownership.

Figure 17 depicts the procedure of two federates passing the ownership of a specified object from one to the other by using *pull/push* approach. *Pull* and *push* approaches for transferring attribute ownership may be applied in different scenarios. For instance, when a customer wants to view merchandise, he/she needs to request the ownership of the object from the vendor by using the "*pull*" scheme, before he/she takes it from shelves whose owner is the vendor. After that, he/she can put the object back onto the shelf by "*pushing*" back the ownership to its original owner.

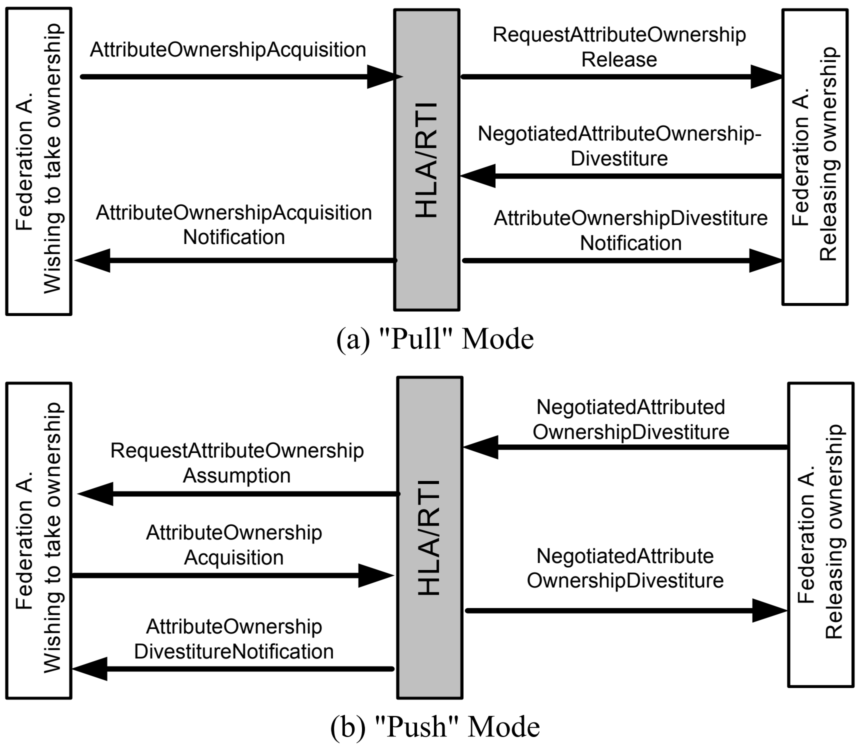

Federation A. Wishing to take ownership
AttributeOwnershipAcquisition
HLA/RTI
RequestAttributeOwnership Release
Federation A. Releasing ownership
NegotiatedAttributeOwnership-Divestiture
AttributeOwnershipAcquisition Notification
AttributeOwnershipDivestiture Notification
(a) "Pull" Mode
NegotiatedAttributed OwnershipDivestiture
RequestAttributeOwnership Assumption
AttributeOwnership Acquisition
NegotiatedAttribute OwnershipDivestiture
AttributeOwnership DivestitureNotification
(b) "Push" Mode

Figure 17. Ownership Interaction Diagrams

3.2. Multi-Agent System in e-Commerce

In the context of a public virtual mall with multiple independent vendors, our goal is to create a software entity (which we call Sales Agent or SA) that will represent each of these vendors. More specifically, we are interested in enabling the consumer to negotiate the price of a commodity by making a binding offer to the SA. The SA is able to respond to such consumer's offer by making use of decision rules based on private information (such as cost prices and inventory numbers) not available in the public catalogue of the mall. A response consists of either an acceptance or rejection of the offer. In case of acceptance, the user is notified and the transaction is recorded at the consumer's offered price. In case of refusal, the user is also notified and the SA has the liberty of making a counter-offer or not. If a counter-offer is made and accepted by the consumer, the SA records the transaction at the counter-offer price.

Our architecture is that of a Multi-Agent System (MAS) (Figure 18) [1]. Each vendor in the mall has one instance of a SA. Consumers are represented by User Agents. Each instance of an agent is a separate entity and is running asynchronously in its own process and possibly on its own machine (SAs are assumed to be continuously running). On the client side, a user agent remembers the user's personal user information through the proper authentication. The client queries the multimedia catalog database through CGI scripts. The user agent is responsible for generating the CGI scripts, parsing the responses from database, which is in CGI format as well, and showing the results on a query window.

Communication between agents is completed through a message passing system that uses a broker agent with a name resolution service to register agents. The broker agent is also considered an agent and run in a predefined machine at a predefined port. It is responsible for registering agents and relaying messages to the appropriate agent. Each SA is identified by the unique name of the vendor it represents (e.g. Sears) and each consumer is identified by the unique user ID provided upon registration in the virtual mall.

[1] This part of work is completed by Prof. T. Radhakrishnan and his team at Concordia University

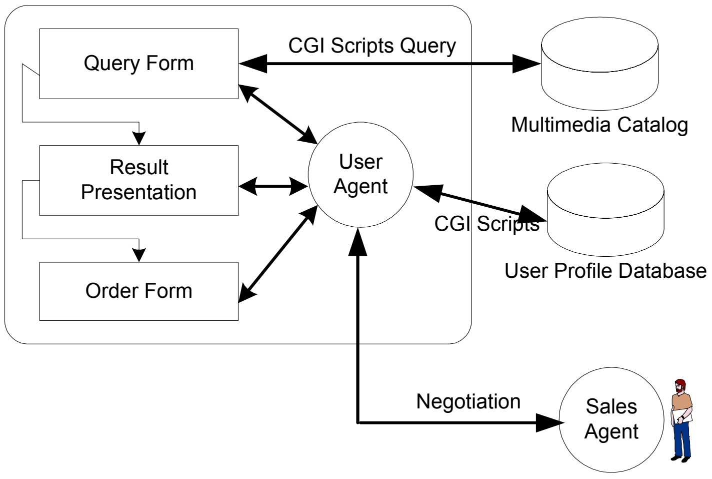


Figure 18. Multi-Agent System in E-Commerce Application

Acknowledgments

The authors acknowledge the research and development contributions of Mojtaba Hosseini, François Malric, Marius Cordea, Prof. T. Radhakrishnan and his team at Concordia University and Ramsey Hage, as well as the financial assistance of Newbridge/Alcatel and CITO. We also acknowledge the financial assistance of the Brazilian Ministry of Education Agency's CAPES scholarship, and the Natural Sciences and Engineering Research Council of Canada (NSERC) Scholarship Program.

References

[1] J. Leigh (1999) A Review of Tele-Immersive Applications in the CAVE Research Network, *Procedings of the IEEE International Conference on Virtual Reality*, Texas, USA..

[2] J. C. de Oliveira; S. Shirmohammadi and N. D. Georganas (1999) Collaborative Virtual Environments Standards: A Performance Evaluation, *IEEE DiS-RT'99*, Greenbelt, MD, USA.

[3] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland (1993) Visually controlled graphics, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6), 602-605.

[4] T.J. Broida and R. Chellappa (1986) Estimation of object motion parameters from noisy images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1), 90-99.

[5] D.B. Gennery (1992) Visual tracking of known 3-dimensional object, *International Journal of Computer Vision*, 7(3), 243–270.

[6] A. Azarbayejani and A Pentland (1995) Recursive estimation of motion, structure, and focal length, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6).

[7] K. Shoemake – Quaternions, *Department of Computer and Information Science, University of Pennsylvania*, Philadelphia, PA 19104.

[8] J. Shi, and C. Tomasi (1994) Good Features to Track, *IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR94) Seattle, WA, USA.

[9] K. Aizawa, T.S. Huang (1995) Model Based Image Coding: Advanced Video Coding Techniques for very Low Bit-Rate Applications, *Proc. IEEE*, 3(2).

[10] X. Shen, R. Hage and N.D.Georganas (1999) Agent-aided Collaborative Virtual Environments over HLA/RTI, *IEEE DiS-RT'99*, Greenbelt, MD, USA.

[11] HLA Homepage, URL at http://hla.dmso.mil/hla

[12] E. T. Powell (1996) The Use of Multicast and Interest Management in DIS and HLA Applications, *Proceedings of the 15th DIS Workshop*, #96-14-125.

[13] jStreaming – http://jStreaming.com

[14] M. D. Cordea, E. M. Petriu , N.D. Georganas, D.C. Petriu, and T. E. Whalen (2000) Real-Time 2½D Head Pose Recovery for Model-Based Video-Coding, *IEEE Transactions on Instrumentation and Measurement*, IMTC/2000 special issue.

[15] Institute of Electrical and Electronics Engineers, International Standard 1516 (2000) IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.