Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Journal of Network and
Computer Applications 27 (2004) 91–111

Journal of
NETWORK
and
COMPUTER
APPLICATIONS
www.elsevier.com/locate/jnca

Distributed virtual reality: building a multi-user layer for the EVE Platform

C. Bouras[a,b,*], Th. Tsiatsos[a,b]

[a]*Research Academic Computer Technology Institute, Greece*
[b]*Computer Engineering and Informatics Department, University of Patras, Greece*

Received 21 April 2003; received in revised form 3 September 2003; accepted 2 October 2003

Abstract

In this paper, we present the design and implementation of a VRML97 multi-user layer, which is introduced in the EVE distributed virtual reality platform. Though other approaches use a VRML node insertion mechanism, such as VSPLUS, or a substitution mechanism, such as SPIN-3D, our approach uses a removal mechanism. Main consideration of our multi-user extension is the ease of transforming single-user virtual worlds into multi-user virtual worlds, as well as the conformity with any standard VRML97 browser. Furthermore, we present the EVE' s communication platform, which can be used in order to support Collaborative Virtual Environments. Main issues regarding the effective network communication as well as the initialization of the 3D scene are discussed in this paper.

Keywords: Collaborative virtual environments; VRML; Distributed virtual reality; Elearning

1. Introduction

Nowadays, the use of Collaborative Virtual Environments (CVEs) is one of the most promising uses of virtual reality. The central concept of CVEs is the one of a multi-user virtual world, i.e. a computer generated space where participants can meet and interact. CVEs might vary in their representational richness from 3D graphical spaces, 2.5D and 2D environments to text-based environments (Snowdon et al., 2001).

Although text based CVEs, such as Multi-User Dungeons (MUDs) and MUDs Object-Oriented (MOOs), provide an interaction paradigm well suited to support a variety of

* Corresponding author. Tel.: +30-2610960375; fax: +30-2610-969016.
E-mail address: bouras@cti.gr (C. Bouras).

1084-8045/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2003.10.002

Virtual Environment (VE) scenarios (Curtis and Nichols, 1994; Brutzman, 1997), this paper is dedicated to 3D based CVEs, motivated by the technological advances as well as the innovative features of 3D based CVEs to support new services such as collaborative work, and distance learning. More specifically, in the past decade, traditional user interface evolved and many VEs emerged due to the rapid growth of the World Wide Web, the availability of powerful 3D graphics accelerators for personal computers (PCs) and high-speed network devices (Picard et al., 2002). Furthermore, using CVEs as communication media, where the users are embodied in the 3D world by an avatar, providing them with a 3D representation that follows the movement of their viewpoint, we can offer the advantage of creating proximity and social presence, thereby making participants aware of the communication and interaction processes with others.

The essence of CVEs is that the shared space defines a consistent and common spatial frame of reference. Furthermore, users' actions and viewpoints are embodied within the space. Therefore, it is possible, at a glance, to visualize what other participants focus on and what their activities are. Consequently, an essential achievement of CVEs is that they combine the participants and the information they access and manipulate in a single common space (Bouras et al., 2000).

From the technical viewpoint, VRML97 (Web 3D Consortium, 1997) is the standard technology for the creation of VEs and for their distribution over the web (Diehl, 2001). This technology has been used by a wide range of VEs and it has been supported by various editors and 3D authoring tools (Bouras et al., 2001). However, VRML does not provide support for multi-user virtual worlds. Thus, there is a definite need for a multi-user extension behind the VRML. This extension should satisfy at least the following requirements: (a) conformity with a standard VRML97 browser, and (b) easy transformation of a single user world into a multi-user 3D world.

The first attempt for standardization of such a multi-user extension has been made by the Web3D's Living Worlds (LW) working group (Web 3D Consortium-Living Worlds Working Group). The main disadvantage of this proposal was its complexity of the multi-user description. Two other approaches have also been proposed: VSPLUS and SPIN-3D. The VSPLUS (Araki, 1998) proposal is a simplification of LW. It offers easier transformation of a single user world into a multi-user 3D world than the LW proposal. However, it works only with predefined multi-user worlds and it does not support dynamic shared objects. The SPIN-3D (Picard et al., 2002) introduces a new approach to VRML data sharing. It supports easy transformation of a single user world into a multi-user 3D world, as well as manipulation of dynamic created shared objects. However, SPIN-3D is not compatible with a standard VRML browser.

The above reasons led us to introduce a new approach to VRML data sharing, which is described in this paper. Main consideration of our multi-user extension is the ease of transforming single-user virtual worlds into multi-user virtual worlds, as well as the conformity with any standard VRML97 browser. Furthermore, we present the EVE's communication platform, which can be used in order to support CVEs. This platform is based on the use of VRML for representing the VE, VRML-EAI (Web 3D Consortium, 1999) for accessing the virtual worlds from external applications, and Java networking for the network communication. Main issues regarding the effective network communication, as well as, the initialization of the 3D scene are also discussed in this paper.

This paper is structured as follows. Section 2 presents the EVE communication platform. Section 3 then discusses the LW, VSPLUS and SPIN-3D approaches for VRML data sharing. This section also describes the VRML data sharing mechanism used in EVE platform. Sections 4 and 5 discuss main technical issues such as initialization and reliable message transmission across the network. Section 6 describes a first prototype for collaborative e-learning, which is supported by EVE communication platforms. Finally, Section 7 presents our vision for the next steps, and Section 8 our concluding remarks.

2. EVE Communication Platform

The EVE communication platform is presented, in this paragraph, in order to introduce the basic modules that are used in the description of the VRML97 multi-user layer, the network communication and the initialization of the multi-user virtual worlds.

As the architecture of the EVE platform is concerned the main different options were (a) a centralized architecture, using client-server communication, and (b) a distributed architecture using peer-to-peer communication (Macedonia and Zyda, 1997; Singhal and Zyda, 1999). The first solution has as its main problems latency and scalability. The main advantage of this solution is that the clients should not have excessive processing power. Also the administration of the clients can be easily achieved in a central way. The second solution requires more processing power from the client's side than the first one. Furthermore, the bandwidth requirements are higher (in case of unicast communication).

Having in mind that our platform aims at supporting CVEs, the main requirements it should fulfil are the following:

- to offer scalability
- to keep the multi-user worlds consistent
- to support specific services for interaction (such as text and audio chat)
- To meet the above requirements, we propose a hybrid multi-server communication model, using two kinds of servers and a database. The two server types are:
- the Message servers, and
- the Application servers.

The basic idea of our architecture (Fig. 1) is to divide the processing load of necessary services of a CVE (such as text and audio chat) into a set of servers aside from users' communication or management of the virtual worlds as described in other models (Bouras et al., 2001).

2.1. Server

The Message server is responsible for manipulating a set of multi-user virtual worlds, serving the latecomers in these multi-user virtual worlds as well as keeping the multi-user virtual world consistent by reflecting every shared event in all participants in the virtual world. It can also back-up another Message server by keeping the current state of

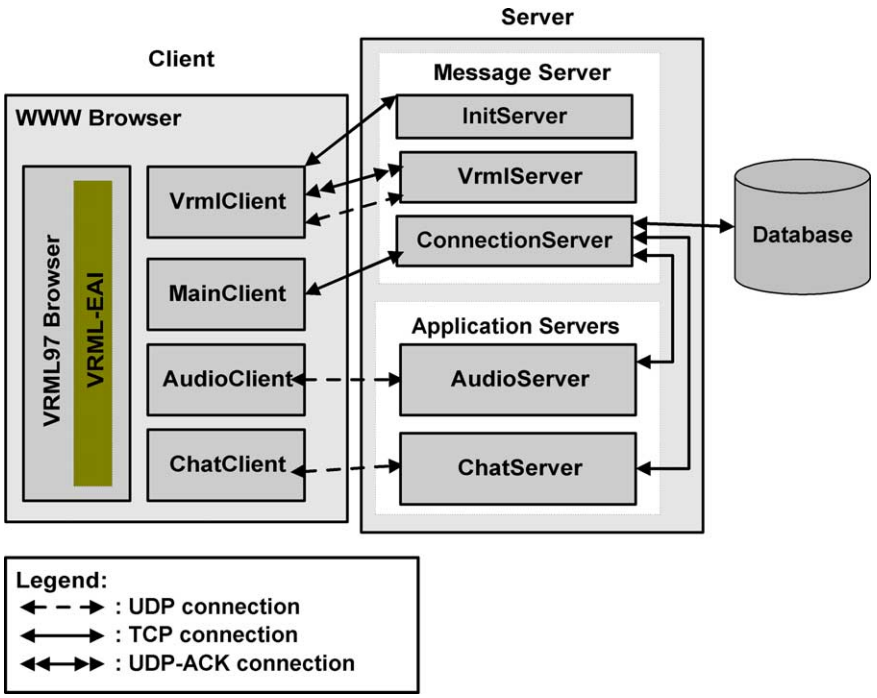

Fig. 1. Architecture and components.

the multi-user virtual worlds of the other server (Bouras et al., 2001) in order to offer scalability and stability. The Message server consists of the following modules:

- ConnectionServer: This server is the core module of the Message server. Every other server is connected to the Connection Server in order to be aware of the users connected to the system. Basic operations of the ConnectionServer are (a) to handle the connection requests of the users in a certain virtual world, (b) to communicate with the database in order to authenticate the users, (c) to accept the users in the system and notify the rest of the servers for the new user.
- InitServer: It is responsible for the initialization of each new client (newly connected user with the system) in the multi-user virtual world. This server holds the last state of the virtual world. When a new user (client) enters the multi-user virtual world, InitServer transmits the current state of the 3D scene to the newly added client.

VRMLServer, which is responsible for sharing the state of the multi-user virtual world as well as for sending update messages regarding the avatar position and orientation in the multi-user virtual world.

The application servers are responsible for offering specific functionality to the participants in the virtual world. Each application server is connected with the ConnectionServer in order to be aware of the connected/disconnected users from

the system. Currently, we have implemented the ChatServer, and the AudioServer. The ChatServer is responsible for text chat communication among the users supporting group chatting (many-to-many) and whispering (one-to-one). The AudioServer is responsible for supporting voice chat communication.

2.2. Database

The database of the system contains information about the users that have access to the system as well as the available groups that a user can join. Furthermore, the database contains the role of each user per group. At that time three roles are supported: Guest, Member and Moderator. The main tables of the database are the following:

- Users: it contains information about the users registered to the system. This information is: the Nickname, Password, and Avatar
- Available_Groups: this table contains all the available groups that are hosted by the platform, and to which a user can be member. The main attributes of the available groups are: ID, Name, and Corresponding 3D world.
- Roles: this table contains info about each user's roles in each group.
- Avatars: It contains information about the predefined avatars that the platform supports.

The database system currently supported is MySQL (MySQL).

2.3. Client

In order to communicate with the above-described set of servers, the client consists of the following components: the web browser, the VRML browser, the MainClient, the VRMLClient, the ChatClient, the AudioClient. The VRML Browser is a plug-in, used for the user's navigation and access to a 3D space (i.e. a VRML world). Any VRML97, EAI compliant browser could be used. The MainClient is a Java applet that establishes the initial connection to MessageServer. It informs the users about the current connection status and about a list of the participants populating the same multi-user virtual world. The VRMLClient is a Java applet responsible for the interaction between the user and the 3D scene. It works in two phases. In the first phase, it is notified about the shared objects of the 3D world from InitServer and initializes the VRML world. This phase is the initialization phase of the 3D scene and it is discussed later in this paper in detail. In the second phase, the VRMLClient sends to and receives events from VRMLServer and updates the multi-user virtual world according to the received events. The ChatClient is a Java applet that implements the exchange of chat messages among users in the same multi-user world and it offers text chat functionality. The AudioClient is a Java applet that records the audio input of the client and sends the audio stream to the audio server. It can also receive and play the audio streams from the audio server.

3. VRML data sharing

The basic goal of a platform for CVEs is the sharing of users' representations and actions in the worlds. Until now there is no standard in this area. The LW working group of the Web3D consortium, which has been frozen, has made the first attempt for standardization. Other remarkable proposals are the VSPLUS proposal, which is a simplification of the LW proposal, and the SPIN-3D approach. These three approaches are presented in the following paragraphs. Then we introduce a new method for VRML data sharing.

In order to show the transformation of a single user world into a multi-user virtual world utilizing the different approaches, we use as an example a single-user VRML world, which contains a simple box that can be rotated in any direction. The VRML code of this world is shown in Fig. 2.

3.1. Related work on VRML97 multi-user extensions

3.1.1. Living worlds

The LW group defined a conceptual framework and specified a set of interfaces to support creation and evolution of multi-user applications in VRML97.

As described in Ref. (Web 3D Consortium-Living Worlds Working Group), LW specification includes fundamental interfaces for building a first generation of shared VRML worlds. The main concepts of the LW proposal for the scene sharing are the SharedObjects and the Multi-User technology (MUTech). The SharedObjects are nodes that contain the description of multi-user objects. These nodes contain the shared behaviour of the object. The MUTech component implements the behavior of shared objects with the use of scripts and Java applets. It provides all network functionalities needed for multi-user interaction. Since it was not in the goals of the LW framework to define a wire protocol, there is no interoperability among MUTechs.

LW specification utilizes scripts and Java applets through the External Authoring Interface (EAI). Therefore, it is designed to work with any standard VRML97 browser. However, the sharing description is blended with the geometry description. Therefore, the creator of the worlds should insert new nodes in the existing VRML97 scene graph and

```
#VRML V2.0 utf8
Group { children [
  DEF box Transform {
    children Shape { geometry Box {}
}
  }
  DEF sensor SphereSensor {}
]}
ROUTE   sensor.rotation_changed   TO
box.set_rotation
```

Fig. 2. An example of single-user VRML world.

```
#VRML V2.0 utf8
# LW external definitions
EXTERNPROTO …
 DEF SO SharedObject {
  private PrivateSharedObject {
   state [ DEF NetRotation NetworkSFRotation {} ]
   visibleDefinition Group { children [
     DEF box Transform {
       children Shape { geometry Box {} }
        }
       DEF sensor SphereSensor {
     }]}
}}
# Removal of the shared ROUTE(s)
# ROUTE sensor.rotation_changed TO box.rotation
# Break of the shared ROUTE(s)
ROUTE          sensor.rotation_changed          TO
NetRotation.set_value
ROUTE NetRotation.value_changed TO box.rotation
```

Fig. 3. An example of multi-user content using Living Worlds approach.

modify the desirable shared routes in order to transform a single-user VRML world into a multi-user VRML world. This has as result a very complex creation of multi-user worlds. It also requires programming skills to carefully update states of a shared object because they generate high network traffic if events are often sent to change them (Araki, 1998).

Fig. 3 shows an example of the sharing of the orientation of a box using the LW framework: the description of the box is placed within the SharedObject description (the bold fonts are used for the code of the single-user VRML world).

3.1.2. VSPLUS

The VSPLUS approach is based on LW specification. As described in Ref. (Araki, 1998), VSPLUS is designed to create multi-user contents from single-user contents providing multi-user extension nodes called NetNodes.

The transformation of a single-user VRML world into a multi-user VRML world takes two steps: (a) selection of the route where the vents pass through and the insertion of the corresponding NetNode, (b) the modification of the routing path to share the events. This procedure for creating multi-user worlds is easier than LW, due to the fact that VSPLUS describes a multi-user content in a more suitable way than with the LW framework. However, VSPLUS is a simplification of the LW results. As a result, VSPLUS does not support the whole functionality that is described in the LW specifications. Furthermore, it works only with predefined multi-user worlds and it does not support dynamic shared objects. Fig. 4, shows how to share the orientation of a box using the VSPLUS syntax (the bold fonts are used for the code of the single-user VRML world).

```
#VRML V2.0 utf8
# VSPLUS external definitions
Group { children [
  DEF box Transform {
    children Shape { geometry Box {} }
  }
  DEF sensor SphereSensor {
  }]}
DEF NetRotation NetSFRotation {}
# Removal of the shared ROUTE(s)
# ROUTE sensor.rotation_changed TO box.rotation
# Break of the shared ROUTE(s)
ROUTE          sensor.rotation_changed          TO
NetRotation.set_value
ROUTE NetRotation.value_ caught TO box.rotation
```

Fig. 4. An example of multi-user content using VSPLUS approach.

3.1.3. SPIN-3D

As described in Ref. (Picard et al., 2002), the SPIN-3D proposal introduces a sharing description for each single-user VRML97 object in order to create a multi-user object. As a result it has to keep the existing VRML97 scene graph unchanged and to use a sharing description for the declaration of the shared nodes (called SharedNodes). More specifically, SPIN-3D proposes a field substitution mechanism rather than a node insertion mechanism (that both LW and VSPLUS use). The main idea behind SPIN-3D is to substitute the object fields that need to be shared by new fields in which a network behavior is embedded (called SharedNodes).

Although SPIN-3D proposal is fine for creating multi-user objects from standard VRML97 objects, the concept of substitution is not applicable to a standard VRML97 browser.

Fig. 5 shows how to share the orientation of a box using the SPIN-3D syntax (the bold fonts are used for the code of the single-user VRML world).

3.2. EVE's Approach

As it can be realized from the previous paragraphs, the afore-mentioned approaches for VRML data sharing have some limitations. LW is complex, VSPLUS does not support dynamic created objects and SPIN-3D requires a proprietary VRML browser. In order to overcome these limitations EVE's approach is introduced (Bouras et al., 2003). The main characteristic of our method is the removal of the ROUTEs (Fig. 6) that we want to share from the single-user world, rather than the 'insertion' proposed by VSPLUS and the 'substitution' proposed by SPIN-3D. These ROUTEs are inserted in a new kind of file: the SVE file.

Benefits from the EVE approach are the simplicity of use, the support of dynamic created objects, and the conformance with a standard VRML97 browser. the drawback is the need for one SVE file per a multi-user 3D world. In the following paragraphs,

```
# Sharing description
SharedSFRotation {
  alias "box.rotation"
}
# VRML 97 Object
Group { children [
  DEF box Transform {
    rotation 0 0 1 0
    children Shape { geometry Box {} }
  }
  DEF sensor SphereSensor {
  }]}
ROUTE sensor.rotation_changed TO box.set_rotation
```

Fig. 5. An example of multi-user content using SPIN-3D approach.

the following concepts regarding the VRML data sharing are introduced: SVE file, SharedNodes, SFCreate and SFDelete Events. Afterwards, we describe how a single-user VRML world can be transformed into a multi-user VRML world using our approach, as well as how we manipulate the sharing of human created objects.

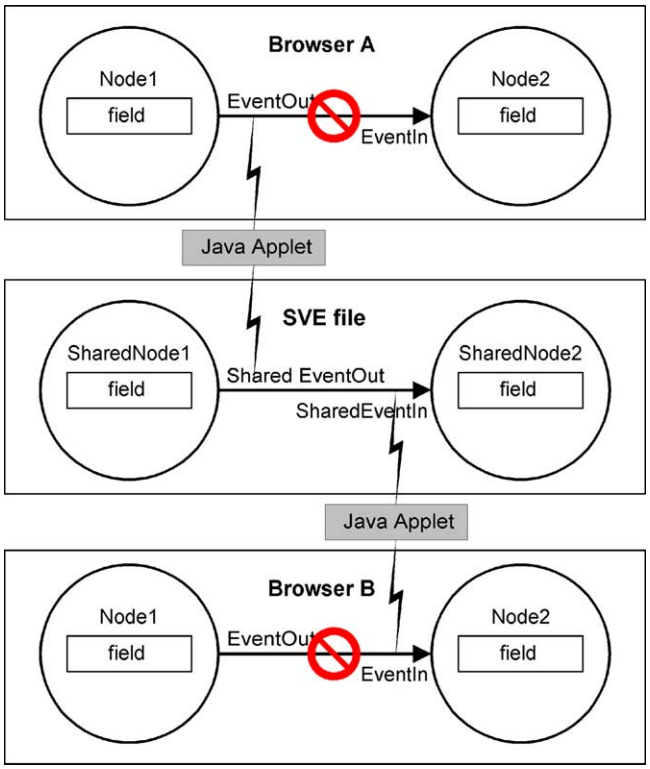

Fig. 6. ROUTE removal.

3.2.1. SVE file

In a multi-user VRML environment, a small part of objects and especially nodes, eventIn(s), eventOut(s) and ROUTE(s) is (are) necessary to be shared. Therefore, each new client in the virtual scene should be aware of both the shared objects and the current state of each shared object (especially if the client is not the first one that entered the VE). The concept of SVE file is introduced in order to: (a) Notify the clients what the sNodes are (sNodes are called the nodes of the single-user virtual worlds, that we want to share); (b) Help the initialization process, in order to keep the latecomers in the virtual world consistent with the previous actions in the 3D scene; (c) Support dynamic created objects; (d) Support necessary actions in a CVE such as text insertion, which is not supported by VRML.

Actually, the SVE file has a VRML-like syntax and it contains all the sNodes and ROUTEs of a multi-user virtual world. More specifically, the SVE file must contain all the ROUTEs that have been removed from the single-user world along with their source and destination nodes. A sNode's definition in the SVE file should not include all the eventIns, eventOuts and fields supported by this node in the single-user VMRL file, but just the eventIns or eventOuts that are involved in a shared route.

Some special nodes are also supported:

- text: a node that 'implements text' is used for the communication of the world with the client in order to ask for and receive text with the help of a Java pop-up window. We decide to introduce this node, due to the fact, that text insertion in a virtual world is a common problem in VRML (Web 3D Consortium-User Input Working Group). Although some effort has been done in order to overcome this difficulty, the available solutions are still far from functional. The EVE platform proposes a different solution through the Java-EAI interface. The user can define a special node in the SVE file that will be able to contact the Java applet through the EAI in order to ask for and receive text. This is achieved with the use of an eventOut of type SFBool and an eventIn of type SFString that are also defined in the SVE file. The eventOut is monitored by the EAI. When its value is TRUE, the EAI pops up a dialog window and asks for text. Afterwards, this text is sent back into the VRML world through the eventIn.
- chair: a node that 'implements chair' is used as a chair inside the world. An avatar can sit on such a node by changing into sit down position and be placed over the node. The position and orientation of the sitting avatar are defined as fields of this node.
- routeCreator: a node that 'implements routeCreator' is used for the communication of the world with the client in order to dynamically create and delete shared nodes and routes.

Apart from these special nodes, two special event types are supported. These event types are SFCreate and SFDelete and are presented later in this paper.

3.2.2. SharedNodes

Using the SVE file, both clients and servers are aware of the sNodes (the nodes that the creator of the virtual world wants to share). Therefore, in order to share the events and

the state of each sNode we need a method in order to update their state. For this reason, we introduce the concepts of SharedNode, SharedEventIn and SharedEventOut.

Actually, a SharedNode (SharedEventIn or SharedEventOut) is the representation (of an instance) of a sNode in Java. Therefore, transmitting all SharedNodes to the client, it solves the problem of handing over the set of sNodes. Furthermore, a SharedNode (SharedEventIn or SharedEventOut) contains the same values as the corresponding VRML sNode (EventIn or EventOut) and a pointer to that node. So, the network behavior is embedded in the SharedNodes. Thus, when a user alters a value of a VRML node in the 3D scene, the same value of the corresponding SharedNode is concurrently updated. Respectively, when an event-message that arrives from the server (VRMLServer) is applied to a SharedNode, the same event is applied to the corresponding VRML node (sNode), modifying the 3D scene.

Furthermore, a SharedNode contains the parameter LastModified, which maintains the sequence number of the last event that has been applied to it. This parameter is very critical for the initialization procedure as described later in this paper.

The SharedNodes can represent the following VRML node types: SFBool, SFColor, SFFloat, SFInt32, SFRotation, SFString, SFVec3f, SFVec2f, and SFTime.

The SharedNodes can be in one of the following modes:

- Locked: a user locks the shared object (SharedNode) and only he/she can manipulate this object until he/she unlocks it. The other participants in the virtual world can see the actions of the user that manipulates the shared object.
- Unlocked: Every user can manipulate a lockable shared object.
- Not able to be locked: The shared object cannot be locked and therefore every user can manipulate it.

This is implemented by adding the parameter Lock in the specification of SharedNode. In order to denote if a shared object is lockable or not we use the boolean parameter ‘lock’ in the SVE file when declaring a sNode.

3.2.3. SFCreate and SFDelete events

In order to solve the problem of the initialization of dynamic created objects, especially for the latecomers we introduce the concept of SFCreate and SFDelete events. The syntax of these events is the same with the syntax of VRML eventIn and eventOut. They are inserted in the SVE file (and not in the original VRML file). We use SFCreate to denote the event that dynamically creates a VRML object in the original VRML file. Actually, an SFCreate is an SFString, whose each new value creates a new SharedNode at the server (InitServer) as a child to the destination SharedNode. We have chosen SFString type, due to the fact that, this type is the more general event type of VRML. Therefore, using SFString type we can simulate every other type. In order to delete every dynamically created object the SFDelete event is introduced. This event is actually an SFInt32 event. The value of this event defines the position of the dynamic created SharedNode that must be deleted. Furthermore, if the value of the SFDelete event is -1, then all ¬children’ SharedNodes of the destination SharedNode will be deleted.

3.2.4. Dynamic sharing

Sometimes an object should not be shared when the world is loaded but later on. Some other times, an object is not predefined and its definition could not be inserted in the SVE file.

These occasions show that it is necessary to have a way to make a VRML node and a ROUTE shared without accessing the SVE file. This can be achieved by using the connection between the multi-user VRML world and the EVE's server through the Java applet (EVE's client) and the network. A special node that 'implements routeCreator' is used for this communication. An eventOut of type SFString informs the Java applet about the dynamically shared routes. The syntax of this eventOut is the following:

sourceNode_name#eventOut_type#eventOut_name#destinationNode_name#eventIn_name

Moreover, this node can use another eventOut of type SFString to delete shared nodes and routes. The syntax of this eventOut is the following:

node1_name@node2_name@node3_name…or node1_name#event_name@node2_name#event_name…

All the routes that contain either a deleted node or a deleted event are also removed.

Fig. 7 gives an example of a node that is suitable for implementing 'routeCreator' and its SVE file (Fig. 8).

3.2.5. Easy transformation of a single-user 3D world into a multi-user 3D world

The main goal in the designing phase of EVE platform is to offer an, as simple as possible, way of making multi-user a single-user VRML world. Indeed, the transformation of a single-user 3D world into a multi-user 3D world is quite simple using EVE platform.

```
DEF   sharedROUTEsCreator Script {
      eventIn     SFString
      sharedRouteCreateString
      eventIn     SFString
      sharedNodeRouteDeleteString
      eventOut    SFString    routeCrParam
      eventOut    SFString    routeNodeDelParam
      url   "javascript:
      function sharedRouteCreateString(value) {
            routeCrParam = value;
       }
      function  sharedNodeRouteDeleteString(value)
{
            routeNodeDelParam = value;
      }"
}
```

Fig. 7. Example of dynamic sharing.

```
node   sharedROUTEsCreator   implements
routeCreator {
      eventOut SFString routeCrParam;
      eventOut                  SFString
routeNodeDelParam;
}
```

Fig. 8. SVE file for dynamic sharing.

In order to show the transformation of a single user world into a multi-user virtual world using the EVE's VRML data sharing we use the previous example of single-user VRML world (Fig. 2).

The creator of the 3D world should follow the following steps:

1. Insertion of both a Collision node (which is the root of the users' avatars) and a Proximity sensor node (which is responsible for giving the position and orientation of the avatars in the virtual world). Using these nodes we can share the users' representation (Fig. 9).
2. Removal of the shared ROUTEs from the original VRML file (in order to add them into the SVE file).
3. Creation of the corresponding SVE file. The SVE file should contain the definition of the nodes that we want to share (sNodes).

More specifically, the starting node of a ROUTE (along with the corresponding eventOut) and the destination node (along with the corresponding eventIn) are inserted. Also, the ROUTEs that are removed from the original VRML file are inserted into the SVE file.

After steps 1 and 2 the VRML world has been transformed as it is shown in Fig. 10 (the bold fonts are used for the code of the single-user VRML world).

After step 3, the corresponding SVE file is the following (Fig. 11):

4. Packet transmission across the network

In our both server-client and server-server communication different types of communication are implemented, each having special characteristics that provide

```
Collision {
  children DEF AvatarRoot Group {
children [ ] }
}
DEF ProxSensor ProximitySensor {
  size 1000 1000 1000
}
```

Fig. 9. Necessary nodes for avatar representation.

```
#VRML V2.0 utf8
# Step 1: Addition of Avatar Root and
Proximity Sensor
Collision {
  children DEF AvatarRoot Group {
children [ ] }}
DEF ProxSensor ProximitySensor {
  size 1000 1000 1000 }
Group { children [
  DEF box Transform {
    children Shape { geometry Box {}
}}
  DEF sensor SphereSensor {
  }]}
# Step 2: Removal of the shared ROUTEs
#ROUTE   sensor.rotation_changed   TO
box.set_rotation
```

Fig. 10. An example of multi-user VRML world using EVE approach.

the requested functionality. In this way, we try to achieve the improvement of the overall performance, as well as to minimize the information that is transmitted across the network.

Generally speaking, the network communication in EVE platform is based on TCP or UDP communication (Fig. 1). Furthermore, we have implemented a UDP with one-way acknowledgements in order to quickly transmit and ensure reliability for a specific class of messages. More specifically, we have divided the messages, according to the actions in the multi-user virtual world, in two classes:

- Position messages (posMes), and
- Important messages (impMes).

PosMes are messages that define the position of avatars in the multi-user world. ImpMes are all the remaining messages. We separate the messages into two categories,

```
node box
{
      eventIn SFRotation set_rotation;
}
node sensor
{
      eventOut SFRotation rotation_changed;
}
ROUTE sensor.rotation_changed TO box.set_rotation;
```

Fig. 11. An example of SVE file.

because posMes have different characteristics and demands from impMes, and therefore different types of communication should be implemented for each type of messages.

The following paragraphs describe in detail the three types of communication that are used in the EVE platform.

4.1. TCP communication

We use TCP communication only when primary objective is the reliable transmission of information. In essence, we use TCP for the server-server (e.g. ConnectionServer-ChatServer) and for the initialization process of the client-server communication. For the first case, TCP is the only choice, as data loss might lead to inconsistency and therefore to problematic execution of the servers. Similarly, when a client is connecting to the server, it needs to receive vital information in order to initialize its 3D scene and any loss of information could lead to unwanted results.

4.2. UDP with one-way acknowledgements (UDP-ACK)

UDP communication is fast but not reliable. On the other hand, TCP is reliable but more time consuming; therefore, a middle solution should be implemented. The idea lies beneath the fact that in some cases packet loss is not very important, yet in others it is catastrophic. To become more specific, in some cases, the lose of a packet that is sent from a client to the server creates no problems, since none of the clients (not even the sender) will receive anything and therefore the consistency will remain. On the contrary, the loss of a packet that is sent from the server to a client could create a severe dependency problem. Thus, for these cases, we use UDP with acknowledgements for the server to client communication (one way). In our platform, we use this type of communication for the transmission of impMes messages, as we need to assure that all clients receive the same impMes messages.

4.3. Simple UDP communication

Simple UDP is the fastest type of communication. It does not guarantee that data will be sent correctly, but that will send them very fast. This type of communication is used when the primary objective is the vivid transmission of packets across the network, and when packet loss does not create any important problems. In our case, we use UDP communication for the transmission of posMes messages. As these messages define the position of avatars into the scene, the only effect that the loss of a message could create, is minor (and temporary) misplace of an avatar in the VRML scene.

4.4. Example of UDP-ACK

At this point, we present the functionality of UDP-ACK communication with an example using impMes messages. Let impMes-1 be the last impMes that has been applied by a client.

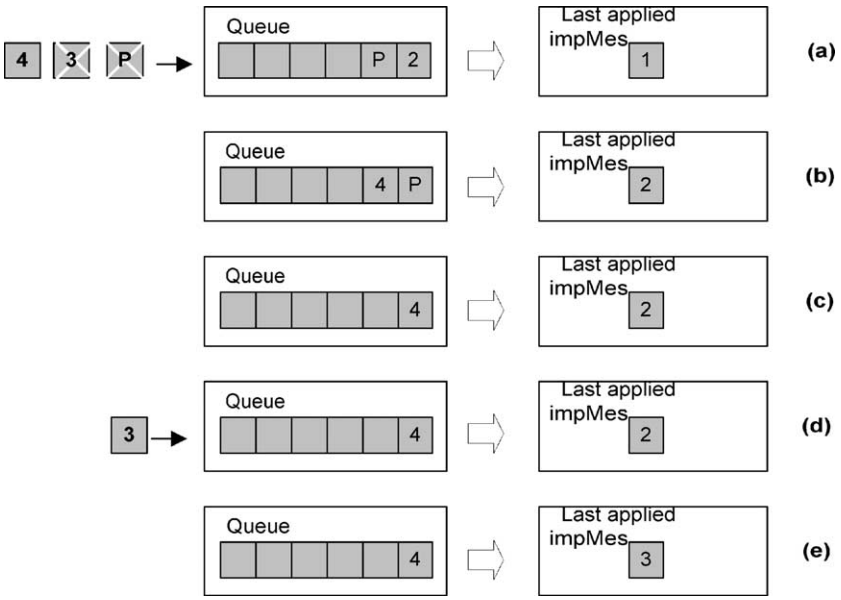

Fig. 12. Example of UDP-ACK.

The client's buffer contains a posMes and the impMes-2, while another posMes, the impMes-3 and the impMes-4 are being transmitted from the server (not yet received by the client). Supposing that, both the last posMes and the impMes-3 are lost in the way, as shown in Fig. 12(a) the client will continue with the execution of impMes-2 (Fig. 12(b)) and the execution of the following posMes until it reaches the impMes-4, as illustrated at (Fig. 12(c)). At this point, the client apprehends the loss only of the impMes-3, ignoring any posMes messages that have been lost, requests its re-transmission from the server and enters a wait mode (Fig. 12(a)). When the impMes-3 is received, it is executed immediately, and then the client continues with the next message in its buffer (Fig. 12(e)).

5. Initialization of the 3D scene

An important issue in a multi-user 3D environment is the initialization process of a new participant. When a new user enters a multi-user world, the shared objects should be updated in his/her client in order to become consistent with all other participants. The client accomplishes this, as follows:

- It is notified about which the sNodes of the VRML scene are.
- It downloads the last state of sNodes, using SharedNodes.
- It applies any updates of the sNodes, which have been made to the sNodes during the initialization period.

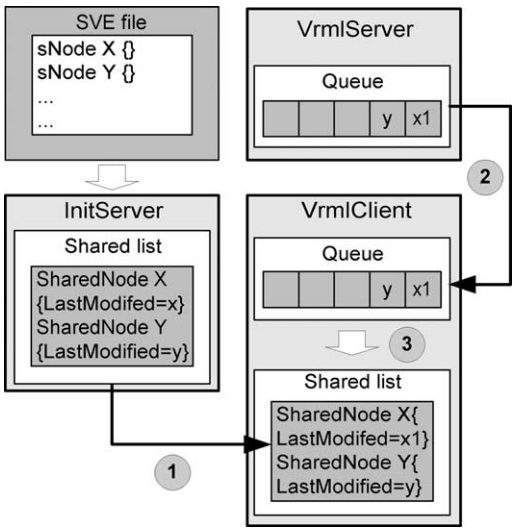

Fig. 13. Initialization process.

The initialization process in the EVE platform is based on the use of both SharedNode and SVE file. When InitServer starts its execution, it parses the SVE file and creates a list of SharedNodes, the SharedList. Then, the initialization process is implemented in the following steps:

- Transmission of SharedList from InitServer: The client opens a TCP connection to InitServer and it receives the list of SharedNodes (Fig. 13(1)).
- Connection to VRML server: When the client starts receiving the SharedList from InitServer, it is concurrently receiving any update messages from the VRML server that has been applied to sNodes by other participants in the VE. These messages are entered in the client's buffer and are not applied until the end of the transmission of the SharedList (Fig. 13(2)).
- Disconnection from InitServer and application of shared events: The client has completed receiving the SharedList and closes its connection with InitServer. Then, it starts applying the events (messages) that are buffered in its queue in the following way (Fig. 13(3)): The client retrieves from the SharedList the SharedNode that the event is supposed to be applied to. Then it compares the SharedNode's LastModified variable with the event's sequence number. If the second one is bigger, then the event is applied to the SharedNode, otherwise it is ignored. For example, as shown in Fig. 13, when the client receives SharedNodes X and Y, their LastModified variables have values x and y, respectively. While downloading the rest of the SharedList, two events are sent from VRMLServer with sequence numbers $x1$ ($x1 > x$) and y that concern SharedNodes s and Y, respectively. After the end of the transmission of the SharedList, the client starts applying the events. First, it compares $x1$-event to LastModified variable of SharedNode X. As $x1 > x$,

the event is applied to the SharedNode and its LastModefied variable is updated to $x1$. On the contrary, when the client compares the value of y-event to the LastMidiafied variable of SharedNode Y, it sees that the event has already been applied to the SharedNode Y, thus it ignores it.

6. Case study: the EVE prototype

CVEs have a good potential to support e-learning services, due to the fact that the can provide the students with an opportunity to experience sensory interactive learning environments, which enable them to move from passive to active learning (Bouras et al., 2001). In addition, such environments are able to support collaborative learning among students at different locations by allowing them to share experiences about exploring a common environment. We call these environments educational VEs (Bouras et al., 2001). The primary goal of an educational VE is to provide tools in order to reproduce conditions that augment interpersonal interaction in a physical educational environment, e.g. a classroom. This goal is effectively satisfied if the educational VE is represented by 3D virtual worlds where the users are represented by human-like avatars. For this reason, the EVE communication platform is exploited in order to support an educational VE. The implementation of such a prototype gives us the possibility to test every aspect of the system in order to offer specific functionality. The environment that we have implemented is called EVE prototype, and it is accessible at http://ouranos.ceid.upatras.gr/vr/. This environment is a simulation of a classroom giving the users communication and collaboration capabilities and necessary tools for realizing collaborative e-learning scenarios.

The virtual classroom is the main area for realizing the learning process. The participants in the virtual classroom have two different roles: tutor (only one participant) and learners. The users, which participate in the virtual classroom, are represented by avatars.

The users' avatars are able to make various types of gestures: expressing opinions (e.g. agree, disagree), expressing feelings, mimics (e.g. happy, sad), as well as showing actions (e.g. move learning content, pick learning content). The virtual classroom is supported by audio collaboration, application sharing and text chat functionality. It also provides a specific place where the users can upload their content and show it to the other participants. This space is a 3D presentation table. Moreover this table offers more functionality such as shared whiteboard, or simulation of a brainstorming board. The user interface of the EVE prototype is depicted in Fig. 14.

7. Future work

The work presented in this paper, is the first phase of the development work of a platform for 3D based CVEs. In order to improve this platform our next steps concern not only technical tasks but also the integration of new functionality into the implemented prototype.

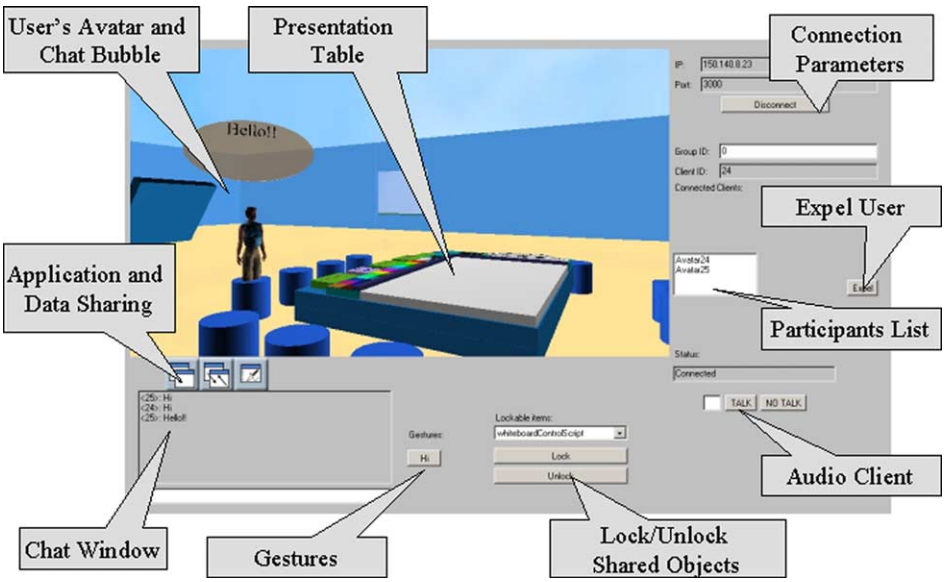

Fig. 14. User interface.

Our first goal is to improve the initialization process for achieving better manipulation of dynamic created shared objects. Furthermore, we would like to implement a VRML parser in order to facilitate the creation of the SVE file. Also, due to the fact that CVEs are increasingly gaining visibility in the field of collaborative e-learning, we are working on creating CVEs that support collaboration and interaction among users for educational purposes. Therefore, from the functional point of view, we would like to enhance our database and to implement a web based interface as a top level of our prototype platform in order to create a CVE, which is focused on providing e-learning services. We are working on creating the web interface for manipulating users (registered as learners and tutors), e-learning courses and educational content. Also new tools will be integrated such as forums and calendar. Furthermore, new types of 3D course rooms will be designed in order to support sub-groups of learners in the virtual classroom. These rooms will be called break-out rooms and will be used in order to realize specific collaborative learning techniques such as think pair share and jigsaw. After the creation of the aforementioned web interface we plan to evaluate the whole environment along with teachers and students of Greek schools.

8. Conclusions

In this paper we present the EVE platform, which can be used in order to support every multi-user VRML virtual worlds.

We also introduce EVE's approach for VRML data sharing. This approach is based on both the concept of the SVE file and the removal of the ROUTEs that we want to share from the single-user world, rather than the 'insertion' proposed by VSPLUS and 'substitution' proposed by SPIN-3D. The concept of the SVE file guarantees the VRML97 conformity, and ensures the proper initialization of the 3D scene in order to serve the latecomers in the multi-user virtual world. SVE file also extends in a transparent way the VRML behavior in order to support dynamic creation of shared objects as well as text insertion in a VRML world. Furthermore, using the EVE's approach, the transformation of a single-user world into a multi-user world is very simple.

The communication architecture of the EVE platform is also presented. Main characteristics of this platform are the efficient communication across the network, which is based on three types of communication, as well as the scalability due to the hybrid multi-server communication model. The usage of Java for creating the servers and clients of the platform as well as the conformity with the VRML standard guaranteed the interoperability and openness of our platform. From the functional point of view our platform provides the basic functionalities in order to support CVEs. These functionalities are text and voice chat, user representation by human-like avatars, support of users' groups, as well as support of users with different roles and rights.

In addition, this paper presents a first prototype, which aims to offer necessary services in order to support collaborative e-learning services. This prototype exploits the features of EVE platform and furthermore simulates a traditional classroom, integrating tools for supporting the learning process, such as 3D whiteboard, brainstorming board, video presenter, drag and drop of learning material, text and voice chat as well as manipulation of objects and users.

References

Bouras C, Philopoulos A, Tsiatsos T. A networked intelligent distributed virtual training environment: a first approach. In Proceedings of fifth Joint Conference on Information Sciences-JCIS'2000—1st International Workshop on Intelligent Multimedia Computing and Networking, Taj Mahal, Atlantic City, New Jersey, USA, February 27–March 3 2000, vol. 2.; 2000. [pp. 604–607].

Bouras C, Triantafillou V, Tsiatsos T. Aspects of collaborative learning environment using distributed virtual environments. In Proceedings of ED-MEDIA, Tampere, Finland, June 25–30; 2001. [pp. 173–178].

Brutzman D. Graphics internetworking: bottlenecks and breakthroughs. In: Dodsworth C, editor. Digital illusions. Chapter 4, Reading: Addison-Wesley; 1997. p. 61–97.

Curtis P, Nichols DA. MUDs Grow up: social virtual reality in the real world. In Proceedings of the IEEE Computer Conference, IEEE Computer Society Press, Los Alamitos California; 1994. [pp. 193–200].

Diehl S. Distributed virtual worlds. Foundations and implementation techniques using VRML, Java, and CORBA, Berlin Heidelberg, Germany: Springer-Verlag; 2001. [ISBN 3-540-67624-4].

Picard S, Degrande S, Gransart C, Chaillou C. VRML data sharing in the spin-3D CVE. In Proceedings of the seventh International Conference on 3D Web Technology-Web3D, Tempe, Arizona, USA; 2002. [pp. 165–172].

Snowdon D, Churchill E, Munro A. Collaborative virtual environments: digital spaces and places for cscw: an introduction, in collaborative virtual environments: digital places and spaces for interaction. In: Snowdon D, Churchill E, Munro A, editors. ; 2001. [ISBN 1-85233-244-1].

Web 3D Consortium. The virtual reality modeling language (VRML)-Part 1: Functional specification and UTF-8 encoding. 1997, http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm.

Web 3D Consortium-Living Worlds Working Group, http://www.web3d.org/WorkingGroups/living-worlds/
Araki Y. VSPLUS: a high-level multi-user extension library for interactive VRML worlds. In Proceedings of the third symposium on Virtual reality modeling language, February 16–20,; 1998. [pp. 39–47].
Bouras C, Psaltoulis D, Psaroudis C, Tsiatsos T. Protocols for sharing educational virtual environments. In Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2001) Split, Dubrovnik (Croatia) Ancona, Bari (Italy), October 09–12, vol. 2.; 2001. [pp. 659–666].
Bouras C, Psaltoulis D, Psaroudis C, Tsiatsos T. Multi-user layer in the EVE distributed virtual reality platform. In Proceedings of Fifth International Workshop on Multimedia Networks Systems and Applications (MNSA 2003) Providence, Rhode Island, USA, May 19–22; 2003. [pp. 602–607].
Macedonia M, Zyda M. A taxonomy for networked virtual environments. IEEE Multimedia 1997;48–56.
MySQL, http://www.mysql.com
Singhal S, Zyda M. Networked Virtual Environments: Design and Implementation. ACM Press; 1999. [ISBN 0-201-32557-8].
Web 3D Consortium. The virtual reality modeling language (VRML)-Part 2: external authoring interface. 1999 http://www.web3d.org/WorkingGroups/vrml-eai/Specification/.
Web 3D consortium-user input working group, http://www.web3d.org/WorkingGroups/kbinput/
Bouras C, Hornig G, Triantafillou V, Tsiatsos T. Architectures supporting e-learning through collaborative virtual environments: the case of INVITE. In Proceedings of IEEE International Conference on Advanced Learning Technologies-ICALT 2001, Madison, Wisconsin, USA, August 6–8; 2001. [pp. 13–16].

Christos Bouras obtained his Diploma and PhD from the Computer Science and Engineering Department of Patras University (Greece). He is currently an Associate Professor in the above department. Also he is a scientific advisor of Research Unit 6 in Research Academic Computer Technology Institute (CTI), Patras, Greece. His research interests include Analysis of Performance of Networking and Computer Systems, Computer Networks and Protocols, Telematics and New Services, QoS and Pricing for Networks and Services, e-learning, Networked Virtual Environments and WWW Issues. He has extended professional experience in Design and Analysis of Networks, Protocols, Telematics and New Services. He has published 150 papers in various well-known refereed conferences and journals. He is a co-author of five books in Greek. He has been a PC member and referee in various international journals and conferences. He has participated in R&D projects such as RACE, ESPRIT, TELEMATICS, EDUCATIONAL MULTIMEDIA, ISPO, EMPLOYMENT, ADAPT, STRIDE, EUROFORM, IST, GROWTH and others. Also he is member of, experts in the Greek Research and Technology Network (GRNET), Advisory Committee Member to the World Wide Web Consortium (W3C), Task Force for Broadband Access in Greece, ACM, IEEE, EDEN, AACE and New York Academy of Sciences.

Thrasyvoulos Tsiatsos R&D Computer Engineer, Computer Technology Institute (CTI) - Research Unit 6 Thrasyvoulos Tsiatsos obtained his Diploma, his Master's Degree and his PhD from the Computer Engineering and Informatics Department of Patras University (Greece). He is currently an R&D Computer Engineer at the Research Unit 6 of Computer Technology Institute, Patras, Greece. He is also attending the third year of studies to obtain his PhD degree at the Computer Engineering and Informatics Department of Patras University (Greece). His research interests include Computer Networks, Telematics, Distributed Systems, Networked Virtual Environments, Multimedia and Hypermedia. More particular he is engaged in Distant Education with the use of Computer Networks, Real Time Protocols and Networked Virtual Environments. He has published 5 papers in Journals and 22 papers in well-known refereed conferences. He has participated in R&D projects such as OSYDD, RTS-GUNET, ODL-UP, VES, ODL-OTE, INVITE and EdComNet and he is currently involved in the project VirRAD.