VRML Data Sharing in the Spin-3D CVE

Stéphane Louis Dit Picard, Samuel Degrande, Christophe Gransart, and Christophe Chaillou
Université des Sciences et Technologies de Lille
LIFL - CNRS UMR 8022
59655 Villeneuve d'Ascq cedex, France
louisdit, degrande, gransart, chaillou@lifl.fr

Grégory Saugis
France Télécom R&D - DIH/HDM/NEW
2 avenue Pierre-Marzin
22307 Lannion cedex, France
gregory.saugis@rd.francetelecom.com

ABSTRACT

In this paper, we present the design and implementation of a VRML97 multi-user layer introduced in SPIN-3D, our Distributed Collaborative Virtual Environment. The main consideration of our multi-user extension is the ease of design of multi-user objects from single-user standard VRML97 objects. Any standard VRML97 browser must at least display the single-user content without taking account of the multi-user description. Whereas other approaches use a VRML node insertion mechanism, such as in Living Worlds, our approach uses a substitution mechanism. And beneath our multi-user extension, we use the Common Object Request Broker Architecture to provide a network communication layer for supporting the virtual meetings.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: 3D Graphics and Realism—*Virtual Reality*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*collaborative computing, computer-supported cooperative work*; C.2.4 [**Computer Communication Networks**]: Distributed Systems—*distributed applications*

Keywords

Virtual Reality Modeling Language (VRML), Collaborative Virtual Environment (CVE), Multi-User Technology, Common Object Request Broker Architecture (CORBA), Multicast Communication Platform

1. INTRODUCTION

In the past decades, traditional user interface evolved and many Virtual Environments emerged due to the rapid growth of the World Wide Web, the availability of powerful 3D graphics accelerators for PCs and high speed network devices (ISDN adapters and ADSL modems). At the beginning, users were isolated in these virtual worlds since there was no support for interaction between them. Nowadays, Virtual Environments tend to become some kind of social spaces, where each user can interact with objects as well as with each other for collaborative activities purposes. This leads to Collaborative Virtual Environments (CVEs). CVEs are powerful tools for a wide range of applications: for instance audio/video conference, co-design meeting or distance learning (*e-learning*).

The Virtual Reality Modeling Language version 2.0 [3] (called also VRML97) established a standard for the description of interactive 3D scenes in the World Wide Web. This language becomes one of the key components of many Virtual Environments since it allows users to create 3D interactive contents without particular programming skills. Indeed, many authoring tools can produce VRML97 contents. VRML97 does not currently provide any support for multi-user virtual worlds. The Living Worlds working group [9] tried to standardize different propositions (Blaxxun, Sony, etc) in a unique multi-user extension: it focused on the integration of support for shared virtual worlds using facilities within the VRML97 specification. The set of extensions proposed by Living Worlds to support shared virtual worlds is not easy to use and requires a deep knowledge of VRML97 specification as well as some programming skills. Other approaches such as VSPLUS [1] enhance the results produced by Living Worlds by designing a multi-user technology for non-programmers. In this case simplifications are made in the creation of multi-user contents starting from single-user contents.

Concerning the communication between the CVE instances, a communication platform is defined. We have a communication platform underneath the VRML97 multi-user extension. This communication platform is responsible for data distribution and management [13]: in order to achieve acceptable video frame rates, each browser has a local copy of each shared object and the communication platform synchronizes the state of these duplicated objects. Data are exchanged between the computers that participate in the collaborative session in order to maintain the overall coherence of the duplicated objects. The platform has to provide an efficient network communication mechanism in order to avoid network lag and thus to maintain the interactive feeling in the shared virtual space: in a CVE the feeling of

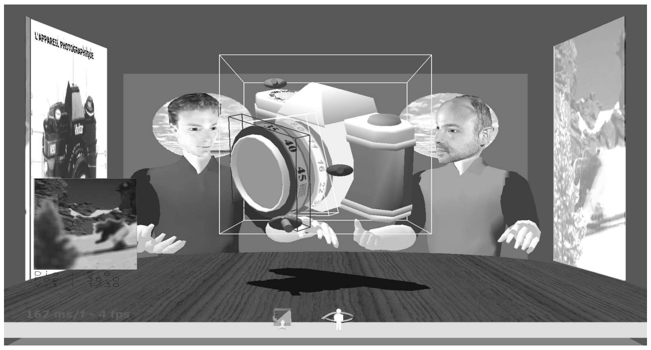


Figure 1: A learning situation in the SPIN CVE.

interactivity is very important in order to understand the work of the other users.
This paper is organized as follows: in a first part, we review the concepts involved in the SPIN-3D CVE; in a second part, we review two VRML97 multi-user extensions, the Living Worlds proposal and the VSPLUS proposal; in a third part, we describe the VRML97 multi-user extension introduced in the SPIN-3D CVE; and finally we present our multicast CORBA based platform used for network communication between different instances of SPIN-3D.

2. CONTEXT: THE SPIN-3D COLLABORATIVE VIRTUAL ENVIRONMENT

In 1994, the project SPACE, funded by France Télécom R&D and the Regional Council of Nord-Pas de Calais (France), aimed to define an interface for small group meetings such as distant learning or co-design situations. A first platform, called SPIN (SPace-INterface), was developed from scratch. Several proposals have been made [5]: a new spatial organization taking advantage of the third dimension, a 3D interaction model using two devices, one for pointing and the other for manipulating objects, and a three step interaction mechanism (select, manipulate and deselect). The objects of the virtual meeting are described with VRML and the application places them in the interface. To support collaboration awareness, users are represented on remote SPINs by clones, namely a 3D representation of themselves. We adopt a "conference table" metaphor: users are located around a table in a single virtual room. Each user organizes the meeting documents inside her/his interface as she/he wants: only documents are shared, not the whole scene. Figure 1 shows a screen shot of the interface of SPIN. The project was presented in the French museum named "La Villette" for one year [10].
However problems arose due to the fact that SPIN uses proprietary standard protocols for communications, proprietary formats for the interaction and for the multi-user description (actually these information are placed in different files), and the system became too complex to program. Then we decided to reorient the project in developing SPIN-3D. Our aim is to propose a complete platform for synchronous collaborative work with which it would be easier to create any collaborative application. We use VRML97 for the description (geometry, interaction and sharing) of the objects. We provide a communication platform for the synchronization of different instances of SPIN-3D and for the management of the virtual place. SPIN-3D needs a special VRML97 browser due to Computer Human Interface requirements (3D pointer for designation, bounding boxes, shadows, and lightning effects missing in a standard VRML97 browser). SPIN-3D is designed for an office usage: as it is impossible to integrate all classical office applications, our platform provides an External Application Interface which allows external applications to access to the objects of the interface. For instance, a user can produce/modify documents with a specialized editor (for example, an HTML document with an HTML editor) and distribute them within the CVE. In a CVE such as SPIN-3D, the different documents evolve during the meeting: the sharing mode of each object can change, it can be either *private* (i.e. the document is presented only in one interface and the user works on it) or *public* (i.e. the document is presented in all interfaces and several users discuss about it and manipulate it).

3. OVERVIEW OF VRML97 MULTI-USER EXTENSIONS

In this section we present two VRML97 multi-user extension: first, the Living Worlds multi-user extension that attempts a standardization of a VRML97 multi-user extension, and second VSPLUS, a simplification of the Living Worlds proposal. Both use the same paradigm: new nodes, which have a network behavior described below, are added into the existing VRML97 scene graph. It is what we call an "insertion" mechanism. Multi-user worlds, that are designed using these two extensions, are designed to work with any standard VRML97 browser.

3.1 The Living Worlds Multi-user Extension

The Living Worlds group defined a conceptual framework and specified a set of interfaces to support creation and evolution of multi-user applications in VRML97 [9]. Since such a framework only uses scripts and Java applets through the External Authoring Interface (EAI), it is designed to work with any standard VRML97 browser. The Living Worlds proposal supports applications which are both:

- Interpersonal: applications support the virtual presence of many people in a single place at a same time.
- Interoperable: multi-user virtual worlds can be assembled from components libraries developed independently by several suppliers.

This multi-user extension provides functionalities for scene sharing. As shown in figure 2, the scene sharing is based on two elements:

- Shared objects: the Living Worlds proposal requires the description of multi-user objects into a node called *SharedObject*. This node contains the shared behaviour of the object via NetworkStates (for example NetworkSFColor, NetworkSFFloat, etc).
- Multi-User technology (MUTech): this 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 Living Worlds framework to define a wire protocol, there is no interoperability among MUTechs.

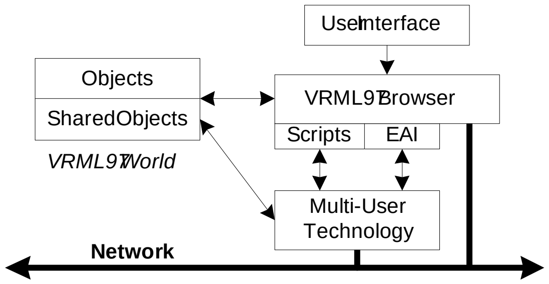
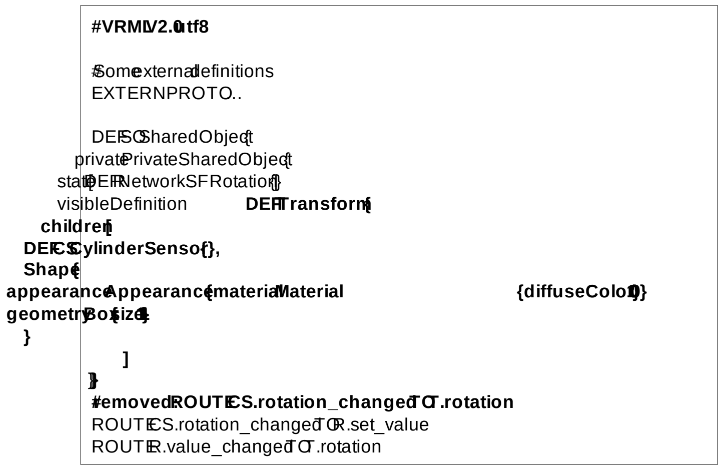


Figure 2: The Living Worlds components diagram.



```
#VRML V2.0 utf8

#Some external definitions
EXTERNPROTO ...

DEF SO SharedObject {
  private PrivateSharedObject {
    state [DEF R NetworkSFRotation {}]
    visibleDefinition DEF T Transform {
      children [
        DEF CS CylinderSensor {},
        Shape {
          appearance Appearance {material Material {diffuseColor 1 0 0}}
          geometry Box {size 1}
        }
      ]
    }
  }
}
#removed ROUTE CS.rotation_changed TO T.rotation
ROUTE CS.rotation_changed TO R.set_value
ROUTE R.value_changed TO T.rotation
```

Figure 3: An example of multi-user content using the Living Worlds syntax ***(in bold, parts coming from the single-user content).***

Figure 3 shows an example of the sharing of the orientation of a box using the Living Worlds framework: the description of the red box is placed within the SharedObject description. Despite the complexity of the multi user description, the world designer adds new nodes into the existing VRML97 scene graph and modifies routes in order to take into account the insertion of the shared state (see figure 4).
The Living Worlds proposal offers interesting concepts, such as the design of a virtual world supported by different MU-Techs or *Zone* and *Pilot/Drone* concepts used in virtual communities.

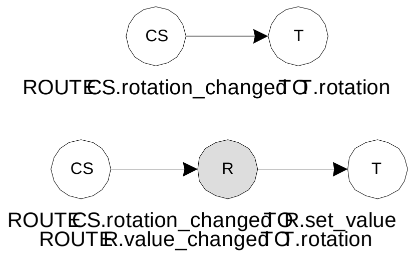

CS
T
ROUTE CS.rotation_changed TO T.rotation
CS
R
T
ROUTE CS.rotation_changed TO R.set_value
ROUTE R.value_changed TO T.rotation

Figure 4: Insertion of NetworkStates in Living Worlds.

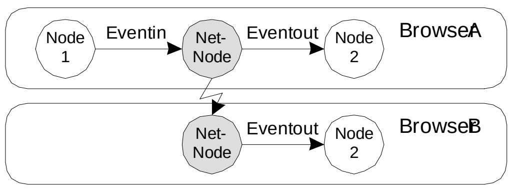


Figure 5: The behaviour of NetNodes.

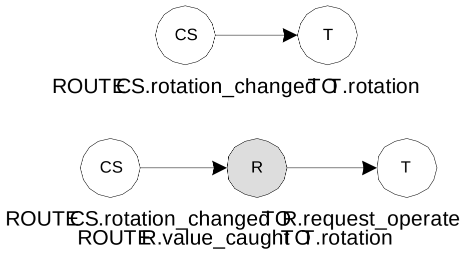

CS
T
ROUTE CS.rotation_changed TO T.rotation
CS
R
T
ROUTE CS.rotation_changed TO R.request_operate
ROUTE R.value_caught TO T.rotation

Figure 6: Insertion of NetNodes in a VRML97 scene graph.

3.2 The VSPLUS proposal

The VSPLUS proposal [1] is a simplification of the Living Worlds results. Therefore it does not offer the range of functionalities proposed by the Living Worlds framework. It is designed for non-programmers and novice people in VRML97, because it offers a simpler description of a multi-user VRML97 content. New nodes, called NetNodes, are added within a single-user VRML97 scene graph. As shown in figure 5, when a NetNode receives an event, it transmits this event to its remote copies by using a centralized mechanism, so that all browsers generate the same *event-out*. For each basic VRML97 type (SFBool, SFColor, etc), a NetNode embedding the network behavior is defined.
As shown in figure 6, the NetNodes are introduced directly in the existing VRML97 scene graph. Existing routes are broken in order to take into account the insertion of NetNodes. Figure 7 shows how to share the orientation of a box using the VSPLUS syntax. Clearly, it allows to describe a multi-user content in a more suitable way than with the Living Worlds framework.
Note that the VSPLUS approach introduces also the notion of *event sampling* for infinite shared event (for example, events out generated by a TimeSensor node) in order to reduce network traffic.

4. THE VRML97 MULTI-USER EXTENSION OF SPIN-3D

In this section, we present the VRML97 multi-user extension introduced for our CVE called SPIN-3D. Our approach is inspired by the work of Living World group and VSPLUS: it allows to describe multi-user contents from single-user contents. Whereas the Living Worlds specification is complex, VSPLUS is an interesting approach except that it works only in predefined multi-user virtual world. In a CVE such as SPIN-3D, the shared virtual world is not predefined: users organize their interfaces as they want and are able to add new objects during the virtual meeting. The new objects can be designed for a single-user usage, and

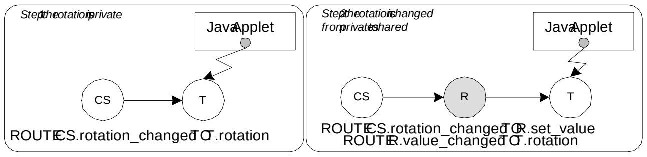
```
#VRML V2.0 utf8

# Some external definitions
EXTERNPROTO ...

DEF T Transform {
  children [
    DEF CS CylinderSensor {},
    Shape {
 appearance Appearance { material Material { diffuseColor 1 0 0 } }
 geometry Box { size 1 }
  ] }

DEF R NetSFRotation {}
#removed: ROUTE CS.rotation_changed TO T.rotation
ROUTE CS.rotation_changed TO R.request_operate
ROUTE R.value_caught TO T.rotation
```

Figure 7: An example of a multi-user content using the VSPLUS syntax *(in bold, the parts coming from the single-user content).*



Step 1: the rotation is private
Java Applet
CS
T
ROUTE CS.rotation_changed TO T.rotation
Step 2: the rotation is changed from private to shared
Java Applet
CS
R
T
ROUTE CS.rotation_changed TO R.set_value
ROUTE R.value_changed TO T.rotation

Figure 8: Dynamic field sharing using the insertion mechanism and external application.

users may want to share such objects. The approach of VSPLUS that consists of the addition of new nodes into the VRML97 scene graph, is difficult to adopt in SPIN-3D. First, when an object is switched from private to public state and vice versa, it requires routes to be modified on the fly. Second, let us consider the situation presented in figure 8: we keep the example of the box introduced before, and we allow a Java applet to modify the orientation of the box. The applet obtains a reference on the rotation field before the sharing. With the insertion mechanism, after the NetNode insertion, modifications made by the user through the applet are only done locally, while we would like them to be propagated to remote interfaces. As shown in figure 9, our approach correctly handles this situation: instead of using a node insertion mechanism (i.e. adding new node in the existing VRML97 scene graph), we propose a field substitution mechanism. The idea is to substitute the object fields that need to be shared by new fields in which a network behaviour is embedded. This notion of substitution is not possible with a standard VRML97 browser. Therefore we defined our own VRML97 browser in which the multi-user technology is included. Our multi-user extension does not offer the complete range of functionalities proposed by the Living Worlds working group. Indeed we discard notions like *Zones*, which are used for space partionning in Virtual Environments with several rooms, in order to limit the bandwith used (the VRML97 browser receives only update messages for shared objects which are in the current user Zone). Nevertheless such a functionality is not required in SPIN-3D as it is designed for small group meetings held in a single room. Our VRML97 multi-user extension is based on the following requirements:

- The respect of the VRML97 syntax, so that standard browsers, such as Cosmoplayer [19] or Blaxxun Contact [2] are able to parse and display correctly a VRML97 single-user part of a multi-user object (without any multi-user support).

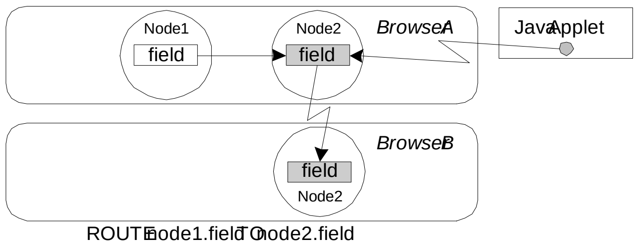


Figure 9: The substitution of fields contained in a node.

- The easiness of creation of multi-user objects from standard VRML97 objects.

- A hierarchical structure of shared data independent of the geometrical description.

- The support of several ways of notification between a local object and its remote copies.

4.1 Sharing Description

For each single-user VRML97 object we introduce a sharing description in order to create a multi-user object. Whereas the proposal of both Living Worlds and VSPLUS is to introduce shared nodes (called *SharedObjects* in Living Worlds and *NetNodes* in VSPLUS) directly into an existing VRML97 scene graph, our approach is to let the existing VRML97 scene graph unchanged and to use a sharing description for the declaration of the shared nodes (called *SharedNodes*). The SharedNodes thus create a sharing tree.

4.1.1 Speci£cation of SharedNodes

SharedNode is the generic name for nodes that are duplicated. Their states must be the same in all instances of the Virtual Environment. In figure 10, we present the generic syntax of a SharedNode. The SharedNode points to a field of the existing VRML97 scene graph whose name is given by the field "*alias*". The value of "*alias*" can be any field of a node named with the DEF mechanism. Then the SPIN-3D VRML97 browser substitutes the field defined by the "*alias*" field by a SharedNode. The SharedNode is shared according to the "*mode*" field. Sharing modes can be "*public*" (seen and used by everybody) or "*private*" (only available locally). The "*how*" field gives information on how the updates must be sent. We identified three possible values:

- "*action*" that indicates an update that must be reliably transported. It corresponds to a discrete action;

- "*flow*" that is used for infinite, real-time, continuous data flows such as video, voice, etc;

- "*animation*" that is used for a flow followed by a reliable state information. Typically, it is useful when an object is moved but only its final position is really important.

```
 EXTERNPROTO SharedXXX [
exposedField SFString alias ""
exposedField SFString mode "public"
exposedField SFString how "action"
 ]"urn:inet:lifl.fr:proto/SharedXXX"]
```

Figure 10: Generic syntax of a shared node.

```
 EXTERNPROTO SharedSFBool [
exposedField SFString alias ""
exposedField SFString mode "public"
exposedField SFString how "action"
 ]"urn:inet:lifl.fr:proto/SharedSFBool"]
```

Figure 11: The syntax of the SharedSFBool node.

SharedNode can be extended in many ways. For example, it may have another value that limits the network bandwidth used. Each new node type, defined as an EXTERNPROTO, corresponds to each field type existing in VRML97. For instance, as shown in figure 11, the SharedNode for sharing a SFBool field is a SharedSFBool. As we use the external prototype mechanism provided in the VRML97 specification for the description of SharedNodes and as the sharing description is placed out of the VRML97 scene graph, browsers which do not support our proposal parse and display the multi-user object using only the single-user object information. Thus we respect our requirements.

4.1.2 Speci£cation of SharedSet: a Sharing Hierarchy

A new SharedSet node will contain a list of shared nodes such as SharedSFBool. This list can also contain other SharedSet nodes to create a hierarchy. In figure 12, we present the syntax of the SharedSet node. The "*mode*" field is useful to disable the sharing of all children of a SharedSet node at once. Actually if the "*mode*" field of a SharedSet node is set to private, it overrides the "*mode*" field of its children.

```
 EXTERNPROTO SharedSet [
exposedField SFString mode "public"
exposedField MFNode children []
eventIn MFNode addChildren
eventIn MFNode removeChildren
 ]"urn:inet:lifl.fr:proto/SharedSet"]
```

Figure 12: The syntax of the SharedSet node.

4.2 Example

In figure 13, we present an example of a multi-user object described with our multi-user extension: it is the same example as before, a box with a shared orientation.

5. COMMUNICATION BETWEEN SHARED NODES

In this section, we explain how shared nodes communicate over the network, i.e. how each SharedNode sends changes to its remote copies. As mentioned in the previous section, a VRML97 multi-user object is defined by the share of its characteristic fields. This means that when an update occurs on one of its fields, the communication platform notifies its

```
 #VRML V2.0 utf8

 # Some external definitions
 EXTERNPROTO ...

 # Sharing description
 SharedSet {
children [ SharedSFRotation { alias "T.rotation" } ]
 }

 # VRML97 scene graph
 DEF T Transform {
children [
DEF CS CylinderSensor {},
Shape {
appearance Appearance { material Material { diffuseColor 1 0 0 } }
geometry Box { size 1 1 1 }
 ] }
 ROUTE CS.rotation_changed TO T.rotation
```

Figure 13: A multi-user content with the SPIN-3D multi-user framework ***(in bold, parts coming from the single-user content).***

remote copies of the update. Concerning the architecture of the platform [12], we have different options:

- A centralized architecture (client/server communications) like in Blaxxun Community Server [2] or in Community Place [8]. This kind of architecture requires a powerful machine for the server process. A client/server architecture introduces latency, which is a problem for interactive feeling: interactions must be done in real-time in order not to disturb users during synchronous collaborative activities. Another problem of such a system is scalability: the performance of the system decreases with a large number of participants.

- A distributed architecture (peer to peer communications) like in DIVE [4] or MASSIVE [7]. Communications are made by using multicast networking. The central server process disappears and each computer is connected to each other. The management of a virtual session is then distributed.

As we want real-time interactions on local computer for local manipulations, the architecture of the SPIN-3D communication platform is distributed. Previous CVE platforms, such as DIVE, developed their own communication platform from scratch (involving sockets programming, messages transport, etc). For example, DIVE system uses a shared memory mechanism for state sharing. Our approach is to use a distributed object computing paradigm to support communications between duplicated objects. In an object oriented context, such a mechanism allows objects to be distributed across an heterogeneous network. These objects may be distributed on different computers throughout a network, living in their own address space outside of an application. However they appear as local objects for the applications which use them. Our idea is to consider a SharedNode as a group of several identical objects (in the meaning of distributed object computing). These identical objects are distributed on the network, leading to a multi-user world composed of several groups. In order to maintain the coherence of the virtual world, we must maintain the coherence of each group of objects. When a change occurs in one of the group copies, the other copies are notified of a state change by using a remote method call.

The three most popular distributed object paradigms are the Microsoft's Distributed Component Object Model [14] (DCOM), the Java Remote Method Invocation [20] (Java RMI), and the Common Object Request Broker Architecture [18] (CORBA). All of them offer approximately the same fonctionnalities. However, DCOM is a prioritary technology of the Microsoft corporation and works only under Windows platforms, and Java RMI is not enough efficient due to the speed of the Java Virtual Machine (JVM). Therefore we chose the CORBA middleware, an open standard that can be used on diverse operating systems as long as there is a CORBA implementation for those platforms.

5.1 Overview of the Common Object Request Broker Architecture

The Common Object Request Broker Architecture (CORBA) is an emerging open distributed object computing infrastructure standardized by the Object Management Group[1] (OMG). Based on the distributed object paradigm, CORBA focuses on providing software components that may be transparently used across network by any application on any platform. CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; parameter marshalling and demarshalling; and operation dispatching. In order to perform these operations, the OMG defines several parts in the CORBA specification (see figure 14):

- The Object Request Broker (ORB) acts as a central object between the data and the application layer. Each CORBA object interacts transparently with other CORBA objects, located either locally or remotely, through the ORB. The ORB is responsible for finding a CORBA object's implementation, preparing it to receive requests, communicating request to it and, if necessary, sending back a reply to the client.

- The Interface Definition Language (IDL) enables the definition of methods that can be invoked for each CORBA object. A CORBA object relies on two stubs generated from its IDL interface to a given language (C, C++, Java, etc). The first one, called stub, is designed for the client side, and provides the illusion that the object is local. The other one, called the skeleton, is designed for the server side (i.e. the object implementation).

- A high level protocol called General Inter-ORB protocol (GIOP) and its implementation over TCP/IP, Internet Inter-ORB Protocol (IIOP) for the definition of messages (invocation request, result message, etc), contains the Common Data Representation (CDR) to map the IDL types to their network representations and the Interoperable Object Reference (IOR) to point to an object.

The CORBA specification provides interoperability between objects implemented with ORB implementations from different compagnies and in different languages across network and operating systems.

[1]The OMG is a consortium including more than 800 members (Sun Microsystems, Microsoft, etc).

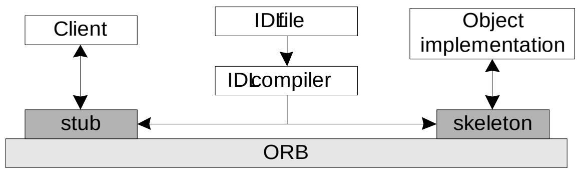


Figure 14: The CORBA specification.

Relying on the previous specification, a distributed application works as follows: to invoke a remote method, the client makes a call to the client side stub. The client stub packs the call parameters into a request message using the IIOP wire protocol. The ORB transport the message and delivers it to the server side stub using the IOR contained in the message. The skeleton unpacks the message and calls the method on the implementation object.

5.2 The SPIN-3D Communication Platform

Because the CORBA specification is an open standard, the specifications evolve. Currently, efficient communication layers are offered for the development of distributed collaborative virtual environments [11] - efficiency both in terms of performance of the system and in terms of software development -:

- A multicast communication layer: as IIOP relies on the TCP/IP protocol, it is impossible to use multicast IP. The OMG thus defines a new implementation of GIOP over UDP/IP called Multicast Inter-ORB Protocol (MIOP) [6, 16]. It provides a group communication by using multicast network communication. Whereas the IOR points to only one object with IIOP, the IOR is a group reference with MIOP. This means that several objects have the same IOR, enabling the invocation of a method on several identical objects in one go (at once). When an object is connected to the multicast ORB, the programmer specifies its group reference (i.e. the group name to which it belongs). Each method call on a multicast reference is executed on each computer which has an object with that reference in memory. Currently, as specified by the OMG, the MIOP communication layer is designed for unreliable communications. However we must keep the consistency between the CVE (i.e. the state of each shared node). Thus, we provide a reliable communication mechanism within MIOP that we call Reliable MIOP (RMIOP) [11].

- A multimedia streaming service [15, 17]: this OMG's standard defines a framework for the development of interoperable distributed multimedia streaming applications. Such a service creates dynamically streams between producers and consumers. For example a microphone can be connected to speakers. The OMG defines the different objects for the management of streams (creation, configuration, destruction, etc) and defines the protocol to set the streams. As in the standard the management of streams is centralized, we adjust the standard to a distributed management using the multicast remote method invocation [11].

The SPIN-3D platform is based on a peer-to-peer system, it thus does not require a server for the management of a

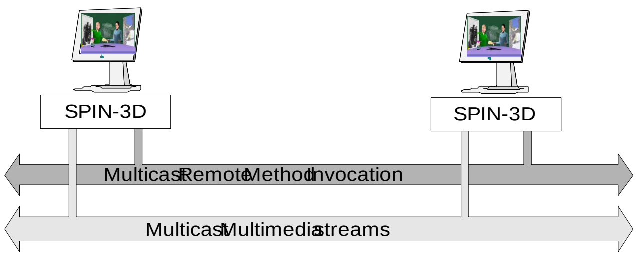


Figure 15: The SPIN-3D communication ways.

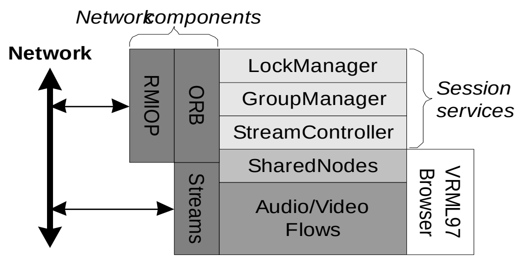


Figure 16: The SPIN-3D core components.

collaborative session. Indeed, multicast networking avoids a centralized management process, and the state of the virtual world is distributed among the different computers which participate to the virtual meeting. Each update made locally on a SharedNode is also made on the remote copies of this SharedNode to maintain the consistency of the world. As shown in figure 15, the communication platform is built over the newest CORBA standards, the Reliable MIOP communication layer and the multicast multimedia streaming service. The SPIN-3D platform distinguishes two types of communication on shared data:

- Discrete actions, for interface control (e.g. button click, etc) and kernel management (e.g. lock manager, session manager, etc). The multicast remote object invocation ensures such a type of communication,
- Continuous actions, for object animations, streaming data such as audio or video, etc. The streaming service allows the creation of streams between an object and its remote copies: the updates pass via the streams.

As shown in figure 16, the SPIN-3D kernel is a compound of several components used for both data sharing and management of a collaborative session (note that in the figure 16, we discard some components of the SPIN-3D kernel: the user interface and the interaction devices management [5] are not represented as this is beyond the scope of this paper):

- The Group Manager manages the session, i.e. the entry or the leaving of participants. We use a reliable multicast protocol defined in [11], we need to know who is connected. A distributed algorithm is used to maintain a list of participants. This is a service which is within each SPIN-3D kernel and is implemented as a CORBA object connected to the ORB, using multicast remote invocations through the RMIOP communication layer.
- The Stream Controller manages the streams established between an object and its remote copies. Like for the Group Manager, the management of the streams is distributed: the stream controller is within each SPIN-3D kernel and is implemented as a CORBA object which is connected to the ORB, and uses multicast remote object invocations.
- The Lock Manager is responsible for the access to shared data. It uses a token mechanism like in DIVE [4], and it ensures that only one SPIN-3D accesses to a SharedNode at a given time. There is one token per SharedNode. When a SPIN-3D wants to manipulate a shared data, it must ask for the token associated with this shared data. The token is requested when the user selects the object; the manipulation starts only if the token is obtained, otherwise the user is notified of the denial. Like for the Group Manager and the Stream Controller, the Lock Manager uses a distributed algorithm for the management of tokens and is implemented as a CORBA object connected to the ORB. The distributed algorithm uses the multicast remote method invocation.
- The VRML97 Sharing Technology: it corresponds to the implementation of the SharedNodes described above. Each SharedNode, such as SharedSFBool, is implemented as a CORBA object and is connected to the ORB. For communication with its remote copies, a SharedNode can use either multicast remote method invocations or multicast streams set on the fly by the stream controller.

5.3 Keeping State Coherence of Shared Data

We saw that each SharedNode is implemented as a CORBA object. For each of them, we describe with an IDL file the methods which can be invoked. The communication platform maintains the state of each SharedNode, i.e. a group of several instances of the same CORBA object.The synchronization of the state of the SharedNode is transparently made by our communication platform. When a change occurs on a SharedNode, the communication platform notifies its remote copies using one shoot update (i.e. using remote method invocation mechanism) or multiple updates (i.e. using the streaming service mechanism). The platform chooses the way of notification by using information contained in the VRML97 description of the SharedNode, namely the information described with the "*how*" field (see the syntax of a SharedNode). This notification is transparent for the SPIN-3D programmers: they only specify this information in the VRML97 description. Moreover they can dynamically modify the notification type by changing this attribute with the mechanisms provided by VRML97 (scripting, routes, etc) and without any call to the communication platform. This is useful for the animations of objects: an object is remotely animated using streaming. Control operations on streams (such as creation, destruction, etc) and data transport are made transparently by the communication platform.

6. CONCLUSION AND FUTURE WORKS

This paper presents the data sharing technology introduced in the SPIN-3D CVE. The VRML97 multi-user extension presented in this paper introduces a new concept. Whereas previous approaches introduce new nodes into the existing scene graph and modify the existing routes in order

to take into account the insertion of these new nodes, our proposal allows to keep the existing VRML97 scene graph unchanged, even concerning the routes, by using the notion of *alias*. Taking advantage of the external prototype mechanism, our multi-user proposal respects the VRML97 syntax. The notion of *node alias* is provided by the VRML97 browser of SPIN-3D. Furthermore, we propose a distributed communication platform in order to maintain the coherence of the shared virtual world. CORBA provides a flexible and efficient solution to support the development of the SPIN-3D platform. The SPIN-3D platform is designed to easily develop collaborative applications.
The SPIN-3D platform is still in development. As SPIN-3D uses a bi-manual interaction model (a 3-DOF pointing device and 6-DOF manipulation device), we are currently working on the definition of new sensors, which allow to use new 3D input devices. We are also working on the concept of plug-in: the kernel capabilities can be extended with the addition of visualization plug-ins. A plug-in is able to display a specific type of information (for example, an HTML formatted document). Another work concerns the visual feedback in the lock mechanism. Indeed, the visual feedback is very important for the understanding of what happens in the interface (when a user obtains or not the access to a shared data).

7. ACKNOWLEDGMENTS

The research project reported here is supported by France Télécom R&D and the regional council of Nord-Pas de Calais (France). We would like to thank Laure France and Laurent Grisoni for proof-reading and improving this paper.

8. REFERENCES

[1] Y. Araki. VSPLUS: A High-Level Multi-User Extension Library for Interactive VRML Worlds. In *VRML'98*, pages 39–47, Monterey, California USA, February 1998.

[2] Blaxxun Interactive. See at http://www.blaxxun.com.

[3] R. Carey, G. Bell, and C. Marrin. ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97). See at http://www.vrml.org/Specifications/VRML97/.

[4] O. Carlsson and O. Hagsand. DIVE - a Multi-User Virtual Reality System. In *IEEE VRAIS'93 Virtual Reality Annual International Symposium Proceedings*, pages 394–400, Seattle, Washington USA, September 1993.

[5] C. Dumas, S. Degrande, G. Saugis, C. Chaillou, M.-L. Viaud, and P. Plénacoste. SpIn: a 3D Interface for Cooperative Work. *Virtual Reality Society Journal*, 1999.

[6] C. Gransart and J. M. Geib. Using an ORB with Multicast IP. In *PCS'99 Parallel Computing Systems Conference Proceedings*, Ensenada, Mexico, August 1999.

[7] C. Greenhalgh and S. Benford. MASSIVE, a Collaborative Virtual Environment for Tele-Conferencing. *ACM Transaction on Computer Human Interaction*, 2(3):239–261, September 1995.

[8] R. Lea, Y. Honda, K. Matsuda, and S. Matsuda. Community place: Architecture and Performance. In *VRML'97*, pages 41–49, Monterey, California USA, February 1997.

[9] Living Worlds Working Group. Making VRML97 Applications Interpersonal and Interoperable. See at http://www.vrml.org/workinggroups/living-worlds/.

[10] S. Louis Dit Picard, S. Degrande, and C. Chaillou. Spin: a 3D CSCW Platform Presented in La Villette. In *First French-British International Workshop on Virtual Reality*, Brest, France, July 2000.

[11] S. Louis Dit Picard, S. Degrande, C. Gransart, G. Saugis, and C. Chaillou. A CORBA Based Platform as Communication Support for Synchronous Collaborative Virtual Environment. In *ACM Multimedia 2001, International Multimedia Middleware Workshop*, Ottawa, Ontario Canada, October 2001.

[12] M. R. Macedonia and M. J. Zyda. A Taxonomy for Networked Virtual Environments. *IEEE Multimedia*, 4(1):48–56, 1997.

[13] B. MacIntyre and S. Feiner. A Distributed 3D Graphics Library. In *SIGGRAPH'98 Conference Proceedings*, pages 361–370, Orlando, Florida USA, July 1998.

[14] Microsoft Corporation. See at http://www.microsoft.com/com/tech/dcom.asp.

[15] S. Mungee, N. Surendran, and D. Schmidt. The Design and Specification of a CORBA Audio/Video Streaming Service. In *HICSS-32 Hawaii International Conference on System Sciences Conference Proceedings*, Hawaii, January 1999.

[16] OMG. Unreliable Multicast Inter-ORB Protocol Request For Proposal. OMG Document orbos/99-11-14, November 1999.

[17] OMG. Control and Management of Audio/Video Streams Specification. OMG Document formal/2000-01-03, January 2000.

[18] OMG. The Common Object Request Broker: Architecture and Specification revision 2.5. OMG Document formal/01-09-01, September 2001.

[19] Silicon Graphics Corporation. See at http://vrml.sgi.com.

[20] Sun Microsystems. See at http://java.sun.com/products/jdk/rmi/.