SharedFantasy: A shared distributed Multi-User Technology using Java's RMI

Oliver Sinnen
INESC-IST
Rua Alves Redol 9, 1000 Lisboa
oli@eniac.inesc.pt

João Leal
INESC-IST
Rua Alves Redol 9, 1000 Lisboa
jleal@se.efacec.pt

João Pereira
INESC-IST
Rua Alves Redol 9, 1000 Lisboa
jap@inesc.pt

Abstract

A Multi-User Technology (MUTech) permits multiple users to navigate and interact in a shared virtual environment. In this paper a WWW based MUTech, called SharedFantasy, is presented based on the open technologies VRML, HTML and Java. For the exchange of shared and distributed information, the Java integrated technology RMI (Remote Method Invocation) is proposed.

After the detailed description of the SharedFantasy functionality, the principles and technology of this new MUTech are discussed in detail. Special attention is given to the Living Worlds oriented VRML structure and to the RMI based communication technology.

Keywords

MUTech, RMI, shared distributed virtual environment, VRML, Java

1. INTRODUCTION

In virtual reality a computer visualises a virtual world, typically modelled in three dimensions, to a user. The interaction and navigation of the user changes the state and the view (the part of the environment visualised to the user) of the world. The "look-and-feel" of such a world is created by the designer. A virtual 3D world can be anything in the range from a spaceship over a house to an outdoor terrain. VRML - an ISO-standard defined by the Web3D consortium [Web3D] - is a language for the specification of virtual environments and the de facto standard for 3D graphics on the WWW.

A multi-user virtual world differs from a simple virtual world in that more than one user interacts with the same world. The users may interact not only with the world, but also with each other (even though this is mostly realised via interaction with the world). Every user has a visual representation in the world called avatar. The avatar typically has the position and orientation of the viewpoint of the user. Other users see the avatar move and rotate as the user explores the virtual world. The avatar's appearance is often human or creature like, for example a woman or a ghost.

A shared virtual environment system is created with the aid of a technology called Muli-User Technology (MUTech). The MUTech is the technology that connects the single-user virtual world technologies to form a multi-user world. All users have their own view of the virtual world, but they are all interacting with the same world. If the users are physically separated, they typically interact with a client-program and a local copy of the shared world. A local action of the user has to be distributed to the clients of the other users for a consistent status of the world. The MUTech of the client observes the world and detects all events and state changes and then sends these to the other clients. Most implementations of MUTechs [Diehl98], [Reitmayr99] use a client-server model for the communication, where every user has one client, which is connected to a server. When the server receives an event from one user it broadcasts the event to all other users (Figure 1).

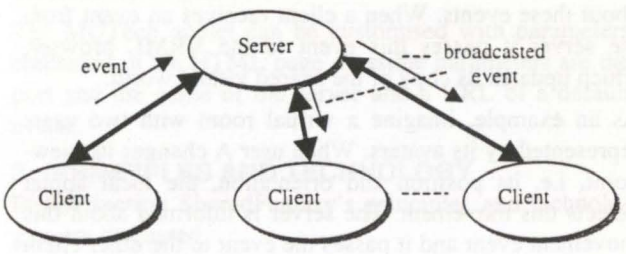

Figure 1 – Client-Server MUTech

The clients and the server are connected over a communication network. For a platform independent and open MUTech the Internet is the preferable network. The Internet is standardised, very widespread and an open technology. Another advantage of the Internet is the possible integration of the MUTech into the World Wide

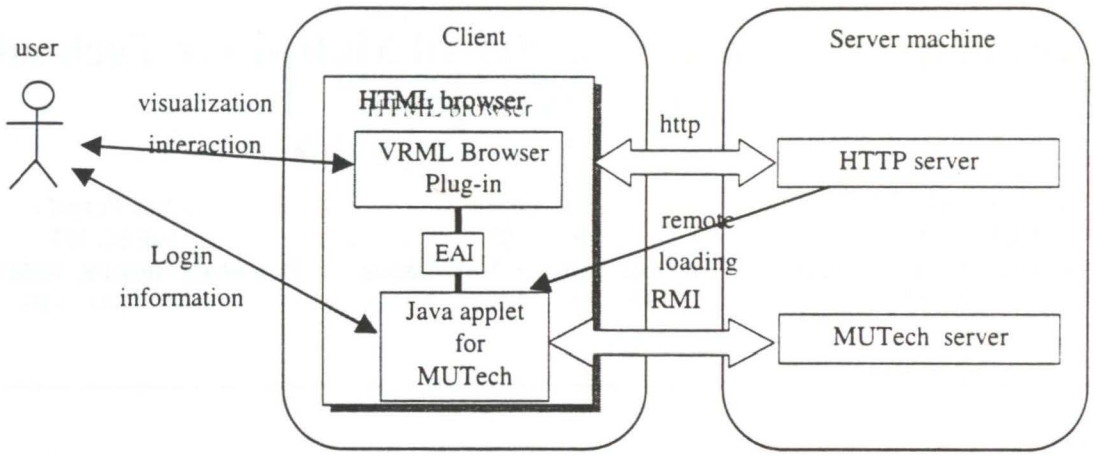

Figure 2 – Client-server structure of SharedFantasy

Web (WWW). A user can enter the virtual world by accessing an HTML page using VRML and Java.

2. SHARED FANTASY

The MUTech system designed and implemented in the context of this article is called SharedFantasy. The aim of SharedFantasy was to create an open, platform independent and simple Multi-User Technology. Therefor, SharedFantasy is Internet (including the WWW) based and uses HTML, VRML and Java as its implementation technologies.

2.1 Client

As the intention is to integrate SharedFantasy in the WWW, the client is based on a Web browser with an installed VRML browser plugin[1] (Figure 2).

The user accesses an HTML page with an embedded VRML world and a Java applet.

The VRML browser is responsible for the visualisation of the virtual world and for the user interaction and navigation.

The Java applet observes the VRML world and detects state changes in the form of events. These events are sent to a central server that in turn informs the other clients about these events. When a client receives an event from the server it passes this event to the VRML browser, which updates its copy of the shared virtual world.

As an example, imagine a virtual room with two users represented by its avatars. When user A changes its viewpoint, i.e. its position and orientation, the local applet detects this movement. The server is informed about this movement event and it passes the event to the other client B. Client B receives the event and sends it to the VRML browser which then updates the client A's avatar position and orientation.

[1] For the development of SharedFantasy Netscape 4.5/6 and Cosmoplayer 2.1.1 were used, but none of the implementation is product specific except for bug workarounds ;-).

The communication between the Java applet and the VRML browser is realised using the External Authoring Interface (EAI). The EAI is a specification [EAIspec] of the Web3D consortium [Web3D] based on the proposal by Chris Marrin [EAIprop].

In the way to contribute for a simple and open implementation of SharedFantasy, the communication between the client and the server uses Java's Remote Method Invocation (RMI). Thus events are exchanged between the client and the server through remote method invocations on Java objects.

2.2 Server

The server machine is composed of two principal parts: an HTTP server and the MUTech server (Figure 2).

The HTTP server is responsible for the static parts of the virtual world. It provides the HTML page with the embedded VRML world and the Java applet, as well as the VRML and the Java classes themselves. Also, the visual representation of the avatars and other shared objects are loaded from the HTTP server. Any available HTTP server (e.g. Apache) can be employed.

The MUTech server manages the state of the virtual world and its visitors. Resulting tasks are the reception and the broadcast of events, the maintenance of a coherent state of the world, and the integration and the removal of new clients to and from the world. As the server uses RMI it is also implemented in Java.

Figure 2 shows the client-server structure of SharedFantasy with the according connections. Note that no communication is necessary between the HTTP server and the MUTech server.

2.3 User Interface

The user interface of the SharedFantasy multi-user world is an HTML page accessible via the World Wide Web (WWW). The page consists of an HTML page with user instructions, an embedded VRML browser and an embedded Java applet, as shown in Figure 3.

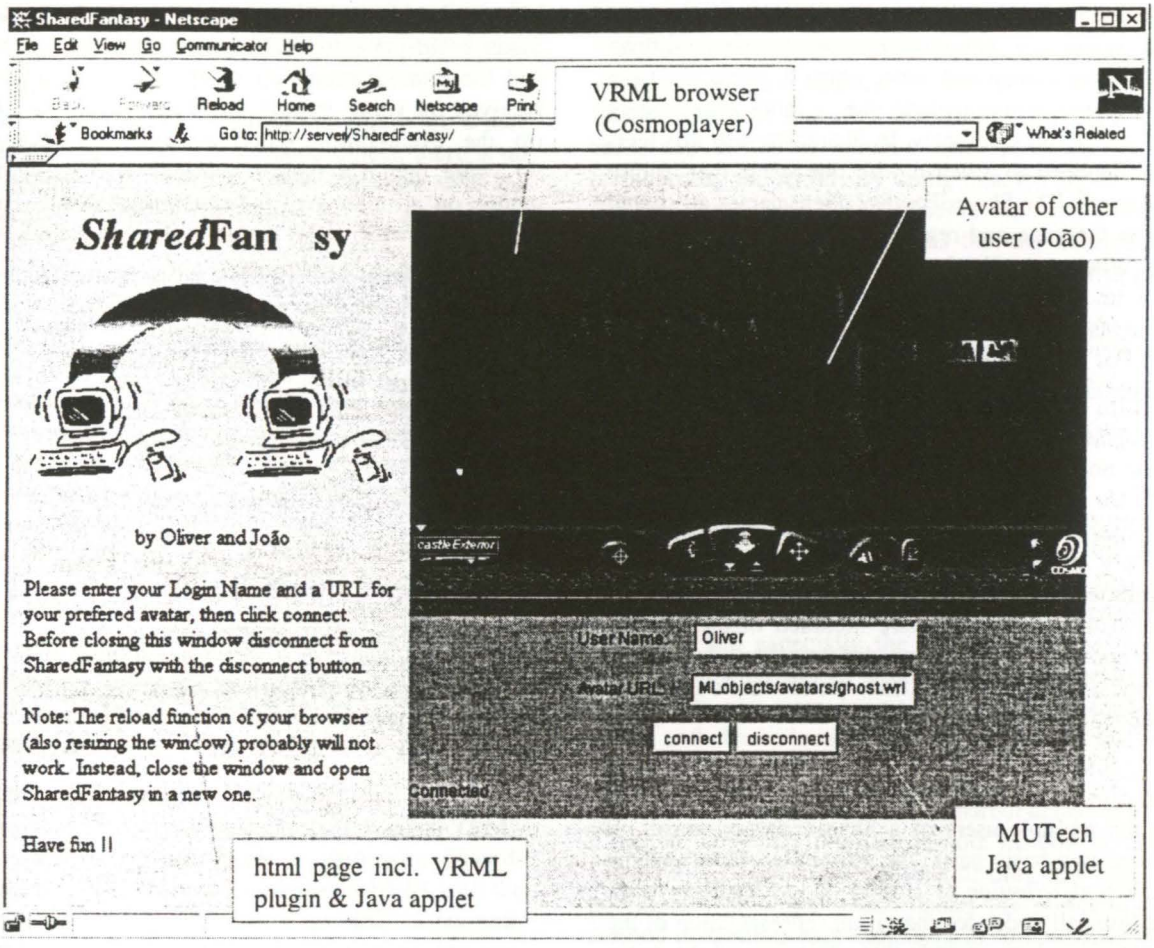

Figure 3 - Screenshot of the SharedFantasy user interface

When SharedFantasy is accessed with a normal URL for the HTML page on the chosen server, the VRML Plug-in and the JVM are started and the shared VRML world and the Java applet are loaded. After the applet is completely loaded, its Graphical User Interface (GUI) becomes visible. To enter the SharedFantasy world the user must connect to the SharedFantasy MUTech server, by entering a login name and by specifying an URL for a VRML file, which will be the visual representation of the user. When the user is connecting to the SharedFantasy world, the state of the world is synchronised with the global state, as maintained by the server, and the avatars of the other users are added to the world.

2.4 Administration

The communication system of SharedFantasy is based on RMI, which in turn is based on the Internet protocols of TCP/IP. Thus, the MUTech server is listening on a TCP/IP port specified as one starting parameter. A second parameter specifies the RMI name for the server object. Both parameters assume default values if they are not defined. Note that, using the default values for the name and the port, the MUTech server does not need any information to be started or maintained. All information the server needs is generated dynamically from the communication with the clients. This includes data about the clients and the shared objects and their current values. The server is not aware of the contents of the VRML file defining the shared multi-user world. The MUTech server must run on the same machine (i.e. same URL) as the HTTP server, since the Java security system only permits network connections of Java applets to the server form where they were loaded. Log information is printed to the standard output.

The MUTech applet can be customised with parameters contained in the HTML page. Possible parameters are the port and the name of the server and a URL of a default avatar.

3. PRINCIPLES AND TECHNOLOGY

In this section SharedFantasy's principles and technologies are discussed.

3.1 Structure of a VRML world

A VRML file of a SharedFantasy world consists of two basic parts: local objects and shared objects. The definition of the local objects is left to the designer and does not need to follow any rules (of course it must be VRML 2.0 conform). The shared objects are all grouped in a *Group* node named "ZONE" (`DEF ZONE Group {}`). This separation of the local objects from the shared objects is oriented towards the *Living Worlds* [Living-

Worlds] specification for Multi-User Technology by the Web3D consortium. *Living Worlds* separates a multi-user world in world, scene and zone, where a world can have multiple scenes, which include one or more zones, being containers for shared objects. In SharedFantasy this hierarchy is flattened to a world with one scene that in turn has one zone[2]. The SharedFantasy client parses all shared objects in the zone and registers the shared states of the objects. When a state is changed locally it is sent over the network to the other clients, while state changes from other clients are passed from the network to the shared objects. For the definition of the shared objects and their states, special nodes were defined.

3.1.1 VRML PROTO Nodes for SharedFantasy

The new node types defined with the VRML construct PROTO for SharedFantasy are complaint with the *Living Worlds* specification. From the complex and comprehensive *Living Worlds* specification some important nodes are supported in SharedFantasy. However, not all fields defined in the specification are included in the SharedFantasy nodes and there is no differentiation between a public and a private part of a node (for example *Zone* and *PrivateZone*).

3.1.1.1 SharedObject

A *SharedObject* is a VRML object whose status is shared among the multiple users of a SharedFantasy world. It consists of two basic parts: the *VisualDefinition* and the *States*. *VisualDefinition* is a kind of a *Transform* node that encloses all nodes for the visual representation of the object. *States* is a group node that comprises all the states to be shared in the multi-user world which are defined as nodes of type *Network*State (explained blow). A *SharedObject* is defined as follows:

```
PROTO SharedObject [
  exposedField SFVec3f position
0 0 0
  exposedField SFRotation orientation
0 0 1 0
  field MFNode visualDefinition    []
  exposedField MFNode states       []
]
{
  Transform {
    translation IS position
    rotation IS orientation
    children IS visualDefinition
  }
  Group {
    children IS states
  }
}
```

3.1.1.2 NetworkState

The *Network*State node represents a single state, and controls its passing back and forth to the network. <State> is substituted with one of the 11 SF and 9 MF VRML data types (e.g. NetworkSFVec3f)[3]. *Network*States build the actual link between the VRML browser and the SharedFantasy client applet. When an event is received on *value_fromnet*, it is sent on *value_changed* (to the event-sink). When an event is received on *set_value*, then it is sent on *value_tonet*. The *tag* must be unique on all clients of the multi user world. The state NetworkSFVec3f, for example, is defined as follows:

```
PROTO NetworkSFVec3f [
  eventIn SFVec3f set_value
  eventOut SFVec3f value_changed
  eventIn SFVec3f value_fromnet
  eventOut SFVec3f value_tonet
  exposedField SFString tag        ""
  field SFBool   localCopy TRUE
]
{
  Script {
    eventIn SFVec3f In IS set_value
    eventOut SFVec3f Out IS
value_changed
    eventIn SFVec3f netIn IS
value_fromnet
    eventOut SFVec3f netOut IS
value_tonet
    field SFBool local IS localCopy

    url "javascript:
      function In(value) {
        netOut = value;
        if( local == true )
          Out = value;
      }
      function netIn(value) {
        Out = value;
      }
    "
  }
}
```

3.1.1.3 SmoothMover

The *SmoothMover* is an implementation of the auxiliary node of the same name defined in the *Living Worlds* specification. The node helps to implement a smooth movement or rotation of shared objects. When one client changes the position of a shared object, all other clients (the remote clients) are informed about this change. However, for a continuous movement the remote clients do not receive all intermediate positions, they rather get the final position of the movement to reduce the number of events sent over the network. To avoid a "jump" of the shared object on the remote clients from the initial to the final position, interpolated positions are generated to simulate the original movement, using the VRML interpolators *OrientationInterpolator* and *PositionInterpolator*. In SharedFantasy the *SmoothMover* is limited to shared objects which are only controlled by one fixed client (in terms of Livings Worlds, the *pilot* of the shared object belongs to one fixed client), which is primarily the

[2] The internal structure of SharedFantasy, however, permits an easy enhancement to the utilization of multiple zones.

[3] SharedFantasy currently supports the *Network*States for *SFVec3f*, *SFRotation*, *SFTime* and *SFBool*.

case with avatars (see below for avatar). The *SmoothMover* definition is omitted here due to space limitations.

3.1.1.4 Avatar

An avatar is a special shared object that represents a user on the remote clients. As it is a shared object, there is no need to define a new PROTO node. To work with the SharedFantasy client the definition of an avatar in a VRML file has to comply a certain structure. The avatar, as a shared object, must be a child of the group "ZONE" and the *tag* fields of the position and orientation *Network*States must be named "Avatar_orientation" and "Avatar_position". A visual definition and a *SmoothMover* node are not necessary on the local client and are inserted automatically on the remote clients when the avatar is added to their worlds. To catch the movements of the user a *ProximitySensor* named TRACKER also has to be included in the VRML file.

```
DEF TRACKER ProximitySensor {
      size 1000000 1000000 1000000
}
```

Two ROUTEs build the connection between the *ProximitySensor* and the *Network*States. Apart from the structure and name requirements, an avatar works like any other shared object.

3.2 Distributed system

In the previous section (3.1) the structure of a VRML world was discussed. The VRML structure for SharedFantasy permits the interaction of the Java applet with the VRML browser. Now, the technology and issues concerning the distributed nature of SharedFantasy are analysed.

3.2.1 RMI

An object oriented and transparent communication approach was required to implement the communication between the MUTech server and the existing clients. Several alternatives were considered (e.g. CORBA) but Java's RMI was chosen because it is already integrated in the virtual machine of the Web browsers currently available. Thus the amount of code that has to be downloaded by each client is reduced.

The MUTech server presents a RMI object which provides a public interface that is available for the clients through remote invocations. To exchange information related with network states and avatar creation / destruction, serialisable objects are used as parameters and return values for the remotely invoked methods.

3.2.2 Coherency and Consistency

Each client only communicates directly with the MUTech server using RMI technology. This Java based technology that provides distribution facilities, is implemented on top of TCP/IP, thus guaranteeing the delivery of messages and also the order of delivery (these are characteristics of the TCP/IP protocol).

So the events used to establish the communication between the server and the client are preserved in their order of emission, and no event will be lost if no failure occurs. If a communication failure happens the sender will be immediately aware of that fact.

Another important feature of the architecture used in SharedFantasy is that each client only communicates directly with the server, which means that all the events are serialised by the server. By doing this, the server is guaranteeing that all the clients observe the same events and in the same order, which means that the SharedFantasy distributed architecture guaranties a total order of events, and consequently also a causal ordering.

If the clients receive the same events in the same order, and due to the fact that the server and the clients have a deterministic behaviour, it can be said that the distributed system always presents a consistent state.

3.2.3 Network Traffic

In the VRML world a great number of events is generated in consequence of user movement within the virtual scene. If all these events would be communicated to the server with one remote method invocation for each event, the client and especially the server would be overload with a small number of users. A trade-off between the necessity to sent many events for a consistent state of the world and the reduction of network traffic had to be found.

The first step of the solution for SharedFantasy is to pack a number of events into one remote method invocation. Observe that each event consists of only few bytes (e.g. coordinates or time), which makes the overhead of the method invocation very expensive in comparison with the data transfer time.

The second step is to reduce the number of events sent with a remote method invocation to one for each *Network*State. When more than one event is generated for one *Network*State between two remote method invocations only the most recent one is sent.

The client of SharedFantasy invokes a remote method on the server a number of times per second as determined by a parameter (currently set to 10). Of all events occurred since the last method invocation, the most recent one is sent for each *Network*State.

To avoid jumps in the animation of shared objects, an interpolator was designed and implemented in the client (more specifically with the VRML) that guaranties a smooth animation when a shared object must change its position or orientation from a certain value to another, by providing interpolated intermediate values.

3.2.4 Scalability

Note that by limiting the number of events sent by every client due to changes in shared objects, a reasonable number of clients can be accepted in a given zone. The limit of the number of clients in a certain zone could even be adjusted dynamically by the server by inducing the clients to use a smaller event update frequency (see above).

3.2.5 Fault Tolerance

If a certain client system fails, then the server will maintain the information associated with the avatar of that user and consequently all the other clients will include a static representation of that avatar. It is obvious that this situation is not significant because the other users do not bother with a static avatar.

From the user's point of view its client's failure is not serious at all, because the user can simply reconnect to the same virtual world.

3.2.6 Security

In SharedFantasy a shared object cannot be destructed or stolen, unless the world designer intended this functionality. So, at the level of the virtual world there are no security mechanisms necessary.

3.2.7 Usability

A system using SharedFantasy is available through a regular Web browser, requiring only the installation of a VRML browser Plug-in (e.g. Cosmoplayer, available for download in the WWW).

When a client enters the multi-user world, all other necessary information is downloaded to the client. This information comprises the HTML page, the VRML world and the Java applet. The size of the VRML file is in the responsibility of the world designer, which is, however, more restricted by the display capacity of the VRML browser (number of polygons!) than by bandwidth restrictions, unless a lot of textures are used. The HTML page has few kBytes and the Java applet has, due to the utilisation of RMI, whose classes are included in the Web browsers JVM, about 50 kBytes uncompressed. Therefor, a reasonable VRML world (which also can be compressed) can even be accessed using a modem, since the communication between MUTech applet and server is not data intensive.

3.2.8 Technologies

The technologies used for SharedFantasy, are without, exception standard or "de facto" standard Internet technologies.

HTML is used for the page containing the applet and the VRML browser. VRML is the "de facto" standard for 3D graphics in the WWW and Java is the standard Web language. RMI, as a Java integrated technology, is used for the communication between the server and the applet.

On the server side, a standard HTTP server and the MUTech server implemented in Java are employed.

All communication protocols used are based on TCP/IP.

4. CONCLUSIONS

In this paper a Multi-User Technology (MUTech) was presented based on open technologies. For the environment of SharedFantasy a client/server structure based on the Internet, namely the WWW, was chosen. Both the client and the server are implemented in Java and the client is embedded in an HTML page for use with a Web browser. The visualisation of the 3D world is realised with the Internet standard for 3D graphics - VRML - using a VRML browser also embedded in the HTML page.

The implementation of SharedFantasy is oriented towards the *Living Worlds* [LivingWorlds] specification of MUTechs, permitting the reutilization of already created worlds of other *Living Worlds* oriented MUTechs (e.g. Deepmatrix [Reitmayr99],[Deepmatrix]) with few adaptations. In order to accomplish SharedFantasy in a platform independent and portable way the generic object oriented language Java was chosen.

SharedFantasy's communication between the clients and the server is based on the Java integrated technology RMI. With this new approach SharedFantasy benefits from the high level of abstraction and the structure of the RMI technology. In contrast to CORBA, RMI is integrated into the JVM of current Web browsers, reducing dramatically the amount of code to be downloaded [Diehl98].

SharedFantasy offers the basic functions of a *Living Worlds* oriented MUTech. Its structure is clear and simple, allowing future projects based on SharedFantasy to benefit from the existing solution. The code and documentation of SharedFantasy will be made available by the authors as Open Source. This way other programmers and designers are free to adapt or optimise SharedFantasy for new requirements.

1.1 Future Work

To achieve scalability beyond certain limits, several zones should be used, which would isolate the clients located in one zone from the other clients. Thus events would be exchange only among the clients within the same zone.

A client failure detection could be implemented in the MUTech server, based on the fact that a faulty client stops to query the server for events. Doing this, the faulty client's avatar information could be removed from the shared world.

Server failures, however, are more serious, but for critical applications an active replication approach could be employed to increase the server's availability.

For commercial or industrial applications (shopping mall, process control) of Shared Fantasy – with the according enhancements of the MUTech – a more sophisticated security model with access control to the shared world and access management for shared objects has to be implemented.

5. BIBLIOGRAPHY

[Cosmo] Cosmosoftware (Cosmoplayer) : http://www.cosmosoftware.com

[Deepmatrix] Deepmatrix: http://www.deepmatrix.com

[Diehl98] S. Diehl, "Towards Lean and Open Multi-User Technologies", Proc. Of the Int'l Symp. On Internet Technology '98, Taipei, Taiwan 1998

[EAIprop] External Authoring Interface (EAI) proposal: http://www.vrml.org/WorkingGroups/vrml-eai/ExternalInterface.html, 1997

[EAIspec] External Authoring Interface (EAI) specification: http://www.web3d.org/WorkingGroups/vrml-eai/, 1999

[LivingWorlds] Living Worlds: http://www.vrml.org/WorkingGroups/living-worlds, 1997

[OMG] Object Management Group: http://www.omg.org

[Reitmayr99] G. Reitmayr, S. Carroll, A. Reitemeyer, M. G. Wagner, "DeepMatrix - An open Technology Based Virtual Environment System", VRML'99, Padaborn Germany, Feb., 1999

[Web3D] Web3D consortium: http://www.web3d.org