In Proceedings of the International Symposium on Internet Technology ISIT'98, Taipei, Taiwan, 1998

Towards Lean and Open Multi-User Technologies

Stephan Diehl

FB 14 - Informatik, Universität des Saarlandes,
Postfach 15 11 50, 66041 Saarbrücken, GERMANY,
Tel.: 0681-302-3915, Fax: 0681-302-3065
Email: diehl@cs.uni-sb.de
WWW: http://www.cs.uni-sb.de/

Abstract

The standardization of the Virtual Reality Modeling Language and its APIs paves the way for platform independent, open standards based implementations of distributed, virtual worlds. After a discussion of key requirements of those and an overview of how the underlying technology moved from proprietary to open standards, we describe our implementation which is based on CORBA, Java and VRML and comment on how open these languages are. Finally, by exploiting CORBA services as far as possible it is our goal to get a lean multi-user technology, which can be loaded over the internet.

1. Introduction

One of the key features which lead to the success of the internet was its openess. Its development was not governed by a company and it was possible for everyone to contribute by submitting proposals to one of the internet task forces. Later on other areas in computer technology tried to copy this process, e.g. the W^3 Consortium or the VRML Consortium were founded to foster and coordinate the development of their respective technologies. There is an increasing awareness that a few big companies have the power to establish poor standards by their huge, dependent customer base. As a consequence, smaller or less successful companies try to survive by forming consortia or organizations to develop common, non-proprietary standards. In this article we will describe a similar observation for multi-user worlds and show how multi-user worlds can be implemented with existing (almost) open standards.

1.1 Virtual Worlds and VRML

Virtual worlds are computer-based models of three-dimensional spaces and objects with restricted interaction. A user can move through a virtual world and interact with those objects in various ways. VRML [8,9] is a file format for the specification of such spaces and objects. A VRML file defines a *scene graph*, by traversing the graph the renderer (i.e. the algorithm which draws the scene onto the screen) computes the appearance, geometry, position and orientation of each object in the scene. VRML was designed to be platform independent and extensible, and it should work with low-bandwidth connections. The major difference to other 3D-file formats is its use of URLs to include spaces and objects over the WWW. At the first WWW conference in spring 1994 a working group on virtual reality interfaces for the WWW agreed that there was a need for a 3D-file format with hyperlinks. Based on Silicon Graphics Open Inventor, Mark Pesce designed version 1.0 of the *Virtual Reality Markup Language*, which was later renamed into *Virtual Reality Modeling Language*. In VRML 1.0 one could only specify static scenes. Interaction was restricted to clicking at hyperlinks. At SigGraph'96 VRML 2.0 was introduced extending the previous standard in various ways including behaviors. Today VRML is broadly used in the virtual reality community and more and more established 3D systems use VRML as a file format to export and import models. In this paper we only consider VRML-based virtual reality browsers. There are other virtual reality browser like SVR or Viscape, but they are of minor importance with respect to the internet.

1.2 Multi-User Worlds

A *multi-user world* is a virtual world, where several users can interact at the same time. These users work at different computers which are interconnected. In multi-user worlds the *avatar* plays a central role. An avatar is the virtual representation of a user. It is put at the viewpoint of the user, i.e., the position in the virtual world from which he looks at the scene. In a single-user world the avatar is only used to detect collision of the user and those objects in the scene. In a multi-user world the avatar is also the visual representation of the user, i.e. it determines how the user is seen by other users. If a user moves his viewpoint, his avatar must also move in the views of the others users. The WWW changed our way to perceive the internet and its services. They are brought to us as a cross-referenced book and we browse through its pages. Multi-user worlds have the potential to change our view again. They provide us with a new metaphor for the internet, its services and its inhabitants (the users). They become objects in space and for fast access the user can tunnel from one point in space to another. One of the major design goals of VRML was to allow for multiple users to act in a virtual world at the same time. So far this goal was not achieved and there is no standard for interaction of several users in a virtual world.

1.3 Requirements

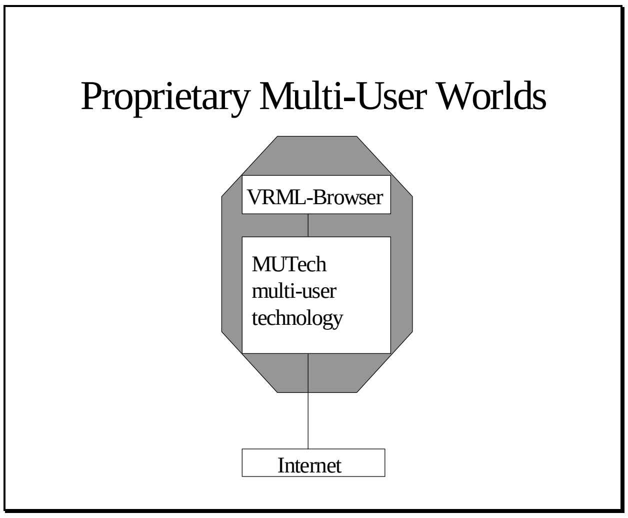
A VRML-browser usually allows two primitive network operations: hyperlinks and inclusion of media stored on different servers in the network. We use the term *multi-user technology* (*MUTech*) for all aspects of network communication in multi-user worlds, which are not provided by the VRML-browser. Essential requirements of MUTechs are listed below, see also [1,2].

- **Adding and Removing Objects**
  If a user enters or leaves the world, or if he adds or removes an object, these changes must be performed in the views of all users. Users and objects must be registered, certain objects might be owned by certain users.
- **Propagation of Changes from Program- and User-Controlled Objects**
  If an object changes its position, orientation or its state in some other way, its new state must be the same in the views of all users. Users may have different rights to change objects.
- **Streaming (Text, Audio, Video)**
  Real-time audio- and video-transmissions, similar to those in phone- and video-conferences, should ease communication among users.

2. Existing Systems

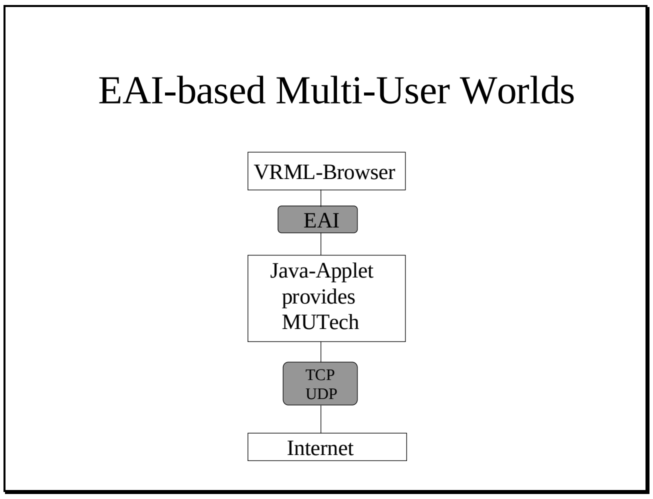
Now we look at different ways to implement multi-user worlds on the internet. We will focus on what languages, interfaces or protocols are used in these implementations.

In the first generation of VRML-based multi-user worlds the different MUTechs were integrated into the browsers of several software companies. VRML was extended by proprietary constructs (node types). Furthermore a proprietary C, C++ or Java API was used to program applications for the multi-users system. Neither the interface between the MUTech and the browser, nor the protocol for inter-browser communication was publicly accessible (Sony's CyberPassage, Blaxxun's CyberSockets). In a first step several companies agreed on an API called Open Community, which was originally developed at Mitsubishi Electric Research Lab (MERL).



The Living Worlds working group of the VRML Consortium is developing a VRML extension, i.e., a set of new node types designed to encapsulate proprietary or open MUTechs. The VRML-browser and MUTech are supposed to communicate only through these nodes.

After some proprietary solutions for applet-browser communication, e.g. using *LiveConnect* and Netscape's Live3D-Plugin [3,10], a working group of the VRML Consortium created a standardized interface called EAI.

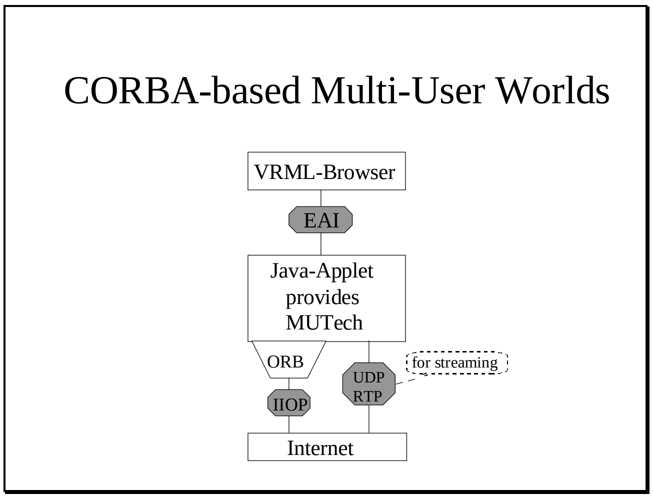


With the *External Authoring Interface* (EAI) it is possible, that an applet in a usual web-browser can access the scene graph of an embedded VRML-browser (as a plugin). Thus instead of using a special-purpose multi-user browser with integrated MUTech, we can now implement a MUTech as a Java-applet in a Java-enabled browser. Such an implementation is described in [1]. It uses two ad hoc wire-protocols (application layer). One is for registering users and objects, and it works on top of TCP. The other is used for streaming changes of objects (movements, rotations) and for text chat, and it works as a modified Real Time Protocol (RTP) on top of UDP.

3. A CORBA-based Approach

CORBA is an architecture for distributed objects in heterogeneous networks and allows objects to mutually access their services. The services provided by an object are specified as interface definitions in the language IDL. These specifications are helpful for the programmer, but also for other objects (dynamic invocation). In the CORBA architecture objects can be implemented in different languages. Above that CORBA offers a variety of services for distributed systems.

A closer look at the MUTech source code in [1] reveals that it contains a lot of administration tasks, which are already provided and more efficiently implemented as services in CORBA. Moreover, if we consider other requirements, which are not covered in [1], e.g. time, security, persistence, then the question arises whether it would be more advantageous to use CORBA. A similar, but more extensive approach is currently pursued in the TeleVirtual Reality (TVR) project at Syracuse University and IBM J.C. Watson Research Center.



As in the previous case the VRML-Browser and the applet communicate via the EAI. Though, for the transmission of time-uncritical messages among browsers we now use CORBA/IIOP. These browsers now communicate through an Object Request Broker (ORB). The programmer does no longer send messages to other hosts, but calls methods of objects, which actually exist at other hosts and that is where the methods get executed. For streaming of text, audio and video IIOP is to slow, as it works on top of TCP. In these cases one might use RTP or another UDP-based protocol.

4. Implementation

Our final goal is a lean implementation of a MUTech, which exploits CORBA services as far as possible. E.g., our current implementation could be simplified by using the CORBA Event Service. Here, by „simplified“ we mean that the source code gets shorter, but it employs more and more complex mechanisms of CORBA. Such a lean implementation could be adapted or optimized for different applications, furthermore as our MUTech is implemented as an applet, every additional line of code means that the user has to wait longer for the multi-user world to start up.

Security in Java-enabled browsers restricts applets to only open connections to the host from which the applet was loaded. For this reason all MUTEchs, which are implemented as applets, use a central server. If we could bypass these security restrictions, and this is possible in Netscape’s Communicator by virtue of the `netscape.security.PrivilegeManager` class, then we could allow direct browser-to-browser communication. For scaleable multi-user worlds with thousands of users this would be a big advantage - just CORBA’s Naming Service might remain centralized. For the above mentioned reason also our implementation uses a central server.

4.1 The Protocol

The protocol between clients and this server is specified by interface definitions in CORBA’s interface definition language (IDL). An interface is a set of signatures. A signature consists of the method name, its arguments and their types as well as the method’s result type. The protocol can be extended by interface inheritance which the reader might know from Java. Via CORBA’s interface repositories and dynamic invocation it should also be possible for two different MUTechs to detect their common methods and cooperate by only using these.

```
module MUTech
{ enum CommandType { changePosition,
                     changeOrientation,
                     ... };
  struct Command { CommandType type;
                   long id;
                   float x, y, z, angle;
                   ...  };
  struct CommandListContainer {
          sequence< Command > commands; };
  interface MUTechCentral {
   string getInitialWorld();
   CommandListContainer
      getUpdates(in long id);
   void sendUpdate(in Command c,in long
id);
   ...
   };
};
```

4.2 The Client

The client, i.e., the applet, accesses the `MUTechCentral` server via a stub, i.e. a special CORBA object. The stub takes care of the methods being executed by the related `MUTechCentral` object on the server. For this purpose the stub calls the clients ORB. The ORB marshals the arguments in the method call and sends them to the server's ORB, the server's ORB unmarshals the arguments, invokes the method of the object, marshals the result and sends it to the client's ORB, which unmarshals the result and returns it to the stub.

For an object to be accessible from a different host, it must be registered under some symbolic name with a CORBA naming server. The client gets a handle to the object from the naming server by providing it with the objects symbolic name. The following two methods of the `MUTechCentral` server object are important. We assume that the server object is bound to the variable `centralRef`:

```
centralRef.sendUpdate(change,myID);
(i)
```

With this method call the modification described by the value of the variable `change` is propagated via the server to all other clients except the one referred to by `myID`, which is the one causing the update.

```
changelist=centralRef.getUpdates(myID);
(ii)
```

The client receives from the server a list of all changes caused by other clients since its last request.

4.3 The Server

On the server the methods of the `MUTechCentral` object are implemented as methods of the class `MUTechCentralServer` as follows:

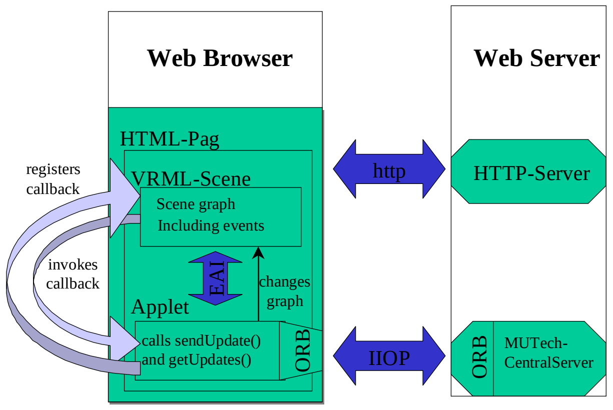
```
public class MUTechCentralServer {
 public void sendUpdate(Command c,
(iii)
                         int id)
        { int i;
          for(i=0;i<clients;i++)
           if (i!=id)

        ((Vector)commands.elementAt(i)).
              addElement(c ); }

public CommandListContainer
   getUpdates(int id)
        { CommandListContainer
           clc = new
CommandListContainer();
         // copies all entries (changes) in
         // commands.elementAt(id) to clc
             return clc; }
          ...
}
```

4.4 Intra-Browser Communication

Now the question arises when to call the above methods. First, via the EAI we can register a callback method with every event in VRML. Events are special field of nodes in the scene graph. Whenever the value of an event changes all registered callback methods are executed.



```
NODE prox=browser.getNode("PROXSENSOR");
ochange
  =prox.getEventOut("orientation_changed");
ochange.advise(this,null);
```

To register the callback method, an object (here: `this`) is passed to the event as an argument of the `advise` method. This object must implement a method with signature `void callback(EventOut, Object)`.

```
public void callback(EventOut event,
                     double when,
                     Object context){
 if (etype==FieldType.SFROTATION)
 { synchronized(this)
   { ev=((EventOutSFRotation) event).
          getValue();
     c=new Command(
        CommandType.changeOrientation,
        myID,
        ev[0],ev[1],ev[2],
        ev[3]-(float) 3.14);
     centralRef.sendUpdate(c,myID);
   }
 }
 ...
}
```

In the above example our callback method will send the current value of the event `orientation_changed` to the server via its `sendUpdate` method. As can be seen from the definition (iii) of the method, the new value is put into a list (an instance of class `Vector`). Finally, it remains to explain how a client receives this list. Again for security reasons an applet is not allowed to listen for requests, so our solution is that the applet launches a thread which periodically issues `getUpdates` requests.

4.4 Restrictions

Our current implementation does not support streaming, we are waiting for JavaSoft's Media API. The development of our implementation was slowed down by problems with the installation of the different software systems involved (e.g. problems with class paths), by their unfinished or insufficient documentation, their bugs (Java's garbage collection frees data which are still accessible via the callback method) and finally by incompatibilities (e.g. Java-IDL and Visigenic's ORB). The source code of our MUTech fits on a few pages. Currently we consider to replace Sun's Java-IDL by the Internet Service Broker (ISB), which is integrated into Netscape's Communicator. The ISB is just Visigenic's ORB in disguise. This integration into the web browser dramatically reduces the number of classes which have to be loaded over the network. As a consequence, it will be possible to load the VRML files of a multi-user world and an application-dependent MUTech directly from the server.

5. Open Standards?

Not all of the open standards mentioned in this article are open in the sense, that everybody could have influenced their development. But they are open in the sense, that everybody could implement them without having to pay any fees (Though, for Java Sun requires that an implementation must be complete and compatible).

Java: In November 1997 Sun eventually did the first step to open the standard [5]. A majority of the ISO members voted for Sun's Java specification to acquire the PAS status (publicly available specification). With this, the process for Java to become an ISO-standard started.

VRML: Since December 15. 1997 a revised version of the VRML 2.0 specification, also known as VRML97, became the International Standard ISO/IEC 14772-1:1997, i.e., an official ISO-standard. Unfortunately this standard does not include the External Authoring Interface. A working group of the VRML Consortium constituted with the goal to add the EAI as an annex to the standard. This means that a VRML-compliant browser should provide the EAI, but it is not mandatory.

CORBA: The Object Management Group (OMG) with over 800 members including all major software vendors develops and standardizes CORBA [6] and its services. Unfortunately all CORBA implementations are source-code incompatible. Fortunately, by virtue of the Internet Inter ORB Protocol (IIOP) ORBs of different vendors can interoperate.

UDP, TCP, RTP: These are internet standards and are described in their RFCs [6], thus they are subject to the open standardization process of the Internet Society.

6. Conclusion

One appealing aspect of the world-wide-web is that different media, programming languages and file formats can be combined [3,11]. Some of these data types are based on open standards. Multi-user worlds can be implemented as such a combination of different systems and languages. We pointed out how the implementations of multi-user technologies employ more and more standardized subsystems and languages. Finally we described our own, CORBA-based implementation in more detail.

References

[1] Bernie Roehl et. al., "Late Night VRML 2.0 with Java", Ziff-Davis Press, 1997

[2] Living Worlds Working Group of the VRML Consortium, http://www.livingworlds.com

[3] Stephan Diehl, "Java & Co - Die Sprachen des Webs: HTML, VRML, Java, JavaScript ", Addison-Wesley, Bonn, 1997 (in German)

[4] James Gosling, Bill Joy, Guy Steele, "The Java Language Specification", Addison-Wesley, 1996

[5] Sun Microsystems Inc., "Java Standardization", http://www.javasoft.com/aboutJava/standardization

[6] Object Management Group, "CORBA 2.0/IIOP Specification", http://www.omg.org/corba/c2indx.html

[7] Request for Comments, ftp://ftp.internic.net/rfc

[8] Stephan Diehl, "VRML", Informatik-Spektrum, Springer, 1997 (in German)

[9] VRML Consortium, "VRML97 International Standard Specification", http://www.vrml.org/Specifications/VRML97/

[10] Netscape Communications Corp., "LiveConnecting Plug-ins with Java", http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/pjava.htm

[11] Yuval Fisher, "Spinning the Web", Springer, 1996