Visual and audio communication between visitors of virtual worlds

MATJAŽ DIVJAK, DANilo KORE
System Software Laboratory
University of Maribor
Smetanova 17, 2000 Maribor
SLOVENIA

Abstract: - The paper introduces some new features into the virtual environments, based on VRML language, like the real time audio streaming, multi-user audio communication and exchange of 3D position and orientation data among visitors of the virtual environment. The audio streams are treated to enable the spatial sound between visitors (observers) of the virtual world. The insertion of all mentioned additional functionality into an arbitrary single-user environment is explained. The solution is carried out in Java language and thus platform independent. The real-time audio communication and representation of visitors with avatars helps to create a feeling of participation in a collective happening.

Key-Words: VRML, avatars, spatial sound, virtual environments, media streaming

1 Introduction

The use of virtual reality (VR) is increasingly popular in many areas, from chemistry, medicine and education to teleshopping. The trend is strengthened by efficiency of modern hardware and software, which enables building virtual worlds with enough realism and applicability to suit most needs. Todays virtual environments (VEs) are usually created using the VRML97 language [1], dont require any specialized equipment and are freely available for all Internet users to visit. The emphasis is usually on the realistic representation of the visual environment, while other aspects of VR such as auditory environment and haptic/kinaesthetic environment are neglected [2]. Since the current VRML standard does not offer language constructs for direct multi-user support [3], most VRML environments today run on a single machine and provide only single-user operation. However, it is believed that the utility of the single-user, standalone systems in the real world is quite limited [2]. Useful VEs should connect people, systems and information streams with each other and support collaboration among a group of users. Most world builders would like to author shared virtual worlds, but developing such environments is a complex and time-consuming task.

The VIRACO (Virtual Reality Audio Conference) system, presented in this paper, is focused on making the task of authoring new shared virtual worlds and converting existing VRML scenes into shared worlds simpler whilst still retaining the ability to work within existing VRML viewers. It is intended to provide basic multi-user support by simply including it in an arbitrary VRML scene. That way, VE authors can create much more useful and interesting models.

The remainder of this paper describes the VIRACO system in detail, with special emphasis on real-time audio communication and graphical presentation of users. The process of including VIRACO in a VRML world is also described.

2 Multi-user virtual environments

The networked virtual environments (NVEs) are often described as systems that permit the users to feel as if they were together in a real environment. Probably the most important aspect of being together with someone, either in the real or in the virtual world, is the ability to communicate [2]. However, the means of natural human communication that are taken for granted in real life, such as speech, facial expressions, lip movement, body postures and gestures are not necessarily supported in NVE systems due to technical difficulties. Most systems have text-based chat capability, and very often the audio communication is supported. Some systems allow a user to select from a few predefined gestures.

Since the VIRACO application is intended to be simple to use, a compromise between functionality and simplicity had to be made. Finally, the
following goals were set: the platform independence of VRML must be preserved and no additional constructs can be added to the language. Visitors of the virtual worlds are represented by simple graphical avatars and no special body gestures are supported. The principal mean of communication is direct audio conversation among visitors, enhanced by spatial sound effects. The number of visitors is of course not limited, although smaller numbers of simultaneous users are anticipated.

One important feature of NVEs that was decided not to be implemented is the use of shared objects and events. Shared objects allow users to manipulate the virtual environment together. For instance, if one user clicks on the switch, a lamp should turn on in every participants local copy of environment. This feature is very useful, but it would greatly increase the complexity of the system. For instance, VE builders would have to identify which objects are shared and which are not, a system for tracking ownership rights would have to be implemented and the initial state of VE would have to be updated with changes from existing environments. For example, if somebody turned on the lamp, a new user entering the world would need to see the light illuminated.

However, we still believe that the system described would provide user with all necessary functionality for immersion in simpler VE.s.

3 VRML97 and its restrictions

Lets check which of the mentioned characteristics are already provided by VRML and which need to be manually implemented.

As already noted, VRML standard does not include any multi-user support so connecting users and transmitting data among them will have to be taken care of by our application.

Situation is only slightly better for the graphical representation of users. While several graphical primitives are defined by the specification [1], no avatar representation exists. Some VRML viewers recognized this weakness and prepared their own solutions. A nice example is the Blaxxun Contact VRML browser, where users graphical embodiment can be turned on or off [4]. Anyway, several human models in VRML are freely available on the Internet and can easily be included in the existing scene. The positions of remote users have to be known, though.

Sound support in VRML is quite advanced. Two nodes are used to add sound to the VRML scene [5]: AudioClip describes the sound source and Sound describes the sound emitter. Addition of spatial effects can be requested using the Sound spatialize field. Sadly, spatial algorithms are not defined by the VRML specification [1]; their implementation is left to the manufacturers of VRML browsers. For our application, the concept of the Sound node is very useful, especially for the sound spatialization. Problems arise only with the AudioClip node. It loads audio data only from the static MIDI and WAV files and does not support streaming media. Because streaming is essential for voice communication, the whole VRML sound model is of no use to us.

In order to keep the ability to view the world in existing VRML viewers, the VRML was extended through the Script nodes [1]. To retain platform independence, Java was used as an external scripting language.

4 The VIRACO system

The network architecture of the VIRACO system is neither fully centralized nor fully distributed; rather it is a hybrid of these two approaches. Its main design is centralized, as shown in Fig. 1, but audio communication occurs using multicasting [6]. Each client represents one visitor of the VE. It is linked with the visitors own copy of the VRML environment, as seen from the visitors point of view. The client program, loaded at startup, provides GUI for controlling the communication, calculates local users position in virtual space and provides graphical representation of users in VE. It also captures sound from microphone and transmits it to other users over a multicast network service. At the same time it reads audio streams from other users, enhances them with spatial sound effects and plays them through the speakers. The server is mainly responsible for maintaining a list of current users and relaying position and control data among the visitors [6].

Fig. 1: Architecture of the VIRACO system
4.1 Audio communication

The basic Java packages do not include support for advanced sound processing, so additional libraries had to be used. We used Java Media Framework API 2.1 (JMF) [7]. JMF provides a unified architecture and managing protocol for manipulating the acquisition, processing and delivery of time-based data. Its plug-in architecture enables programmers to directly access media data and easily customize and extend frameworks functionality. A large number of standard media types can be used, such as AU, AVI, MIDI, MPEG, QuickTime, WAV and others. Capturing of media data can be accomplished using a variety of supported devices and can be used even in applets. For transmission of media streams over network JMF uses Real-Time Transport Protocol (RTP) [8]. RTP provides end-to-end network delivery services for the transmission of real-time data. It is network and transport protocol independent and can be used over both unicast and multicast network services.

In our application JMF is first used to capture users voice from the microphone. The resulting audio stream is encoded into high quality MPEG format and is sent across the network using RTP. When such stream is received, it is first decoded into raw data. Then our custom SpatializeEffect [6] JMF plug-in is used to add 3D information to the sound in real-time. Finally, the audio is played through speakers or headphones.

While present in the virtual environment, the user has the control over the communication process. He can decide when to start/stop the transmission of audio to other users, select which users he wants to listen to and change the volume of individual audio streams. Besides that, detailed statistical information about RTP transmissions can be received, such as number of lost packets, number of bytes received, current bit rate, etc [6].

4.1.1 Spatial sound

The spatial sound effects enhance speakers voice with the positional information. This way, a listener can determine speakers position merely by listening to his sound. This adds a new dimension to realistic experience of the virtual scene. Many methods of establishing spatial sound have been developed, efficient ones being quite extensive and complex [9]. For the use in VIRACO application a SpatializeEffect plug-in has been implemented. Because it needs to work with raw sound samples in real-time, only two simple effects are applied to the sound. Lets describe them more in detail.

It is common knowledge that distant sound sources appear quieter than sources near the listener, due to the occluding effect of the air. So, in order to simulate the distance to the source, the sources amplitude has to be modified. One way of changing the amplitude in dependence of distance is shown in the following equation:

\[
g(d) = \begin{cases} 
1 & : d \leq \text{MIN\_DIST} \\
\frac{d - \text{MIN\_DIST}}{\text{MAX\_DIST} - \text{MIN\_DIST}} & : \text{MIN\_DIST} < d < \text{MAX\_DIST} \\
0 & : d \geq \text{MAX\_DIST} 
\end{cases}
\]

where \( g(d) \) is normalized sound gain, \( d \) is the distance between the sound source and the listener, \( \text{MIN\_DIST} \) is the distance at which the sound is heard at full volume, and \( \text{MAX\_DIST} \) is the distance beyond which no sound is heard. Between \( \text{MIN\_DIST} \) and \( \text{MAX\_DIST} \) the gain decreases linearly.

If the sound source is not directly in front of or behind the listener, sound waves reach one of the listeners ears a fraction of a second before the other. This difference, known as the interaural signal difference (\( \Delta t \)), helps the brain detect the direction of the sound. Its value ranges from 0 (when the source is directly in front of the listener) to approximately 650 \( \mu \)s (when the source is on the left or on the right side) [9]. It can be calculated using the following model:

\[
\Delta t = \frac{D(\varphi + \sin \varphi)}{2c},
\]

where \( D \) is the distance between the ears, \( \varphi \) is the angle between the view direction and the sound source direction, and \( c \) is the speed of sound. Simulation of the users position in space is achieved by storing the appropriate number of one audio channel samples into a FIFO buffer. The samples from the other channel are played undelayed.

4.2 Graphical representation of users

Visiting virtual worlds is much more interesting, if we meet other visitors. In order to see them, they have to be represented by graphical avatars with the correct position and orientation. Unfortunately, this position information is stored internally by the VRML viewer and is unavailable at run-time. To track users movement, a custom navigation interface can be used. This solution is a bit awkward, because VRML browsers already provide
sophisticated and well-known navigation options. Better results can be achieved with a little trick. If a ProximitySensor [5] with the box-shaped active region covering the whole virtual world is constructed, then each users move is visible through position_changed and orientation_changed eventOuts (Fig. 2).

```xml
DEF sensor ProximitySensor {
  size 1000 1000 1000
}
DEF loaderScript Script {
  # ...
  eventIn SFVec3f position
  eventIn SFRotation orientation
  # ...
}
ROUTE sensor.position_changed TO loaderScript.position
ROUTE sensor.orientation_changed TO loaderScript.orientation
```

Fig. 2: Sensing users movements inside the Script node

Once users motion is detected, it is immediately sent to the server and distributed to all remote participants. When the notification of the move is received from the server, the client needs to draw particular users graphical embodiment on the correct position in the VE. VIRACO currently uses only three simple VRML models that were freely available on the Internet (Fig. 3) [10]. Each remote user is assigned one such avatar in a circular (round-robin) fashion. New graphical object is inserted into the existing VRML scene graph using the Browser Script Interface method loadVrmlFromURL [5]. As a result, local user can observe other participants as they appear, turn around and move through the shared virtual world.

```xml
DEF sensor ProximitySensor {
  size 1000 1000 1000
}
DEF userGroup Group {} 
DEF loaderScript Script {
  eventIn SFVec3f position
  eventIn SFRotation orientation
  eventOut MFNode avatars_changed
  # ...
  field SFString serverAddress "164.8.253.96"
  field SFString serverPort "33300"
  field SFString posx "-4.0"
  field SFString posy "1.6"
  field SFString posz "7.0"
  field SFString rot "349"
  field SFString minDist "2.0"
  field SFString maxDist "13.0"
  url "VIRACOLOader.class"
}
ROUTE sensor.position_changed TO loaderScript.position
ROUTE sensor.orientation_changed TO loaderScript.orientation
ROUTE loaderScript.avatars_changed TO userGroup.children
```

Fig. 5: VIRACO loading code

4.3 Usage of VIRACO system

The result of adding VIRACO to a simple virtual room can be seen on Fig. 4. Three users are currently present in the environment: two remote, depicted as a man and a woman, and one invisible local user. The user interface has two main parts. The first one is the VRML browser, where users view of the virtual world is shown and navigation controls are located. The second part is the VIRACO control panel, where current participants are listed and communication control is possible. The control panel is started automatically after the virtual world is loaded.

![Fig. 4: VE with three visitors](image)

![Fig. 3: Avatars in VIRACO](image)
The inclusion of VIRACO system into an arbitrary single-user VRML environment is very straightforward and simple. In order to do that, nodes and routes shown in Fig. 5 need to be added to the top-level VRML file. That way a script that loads the main VIRACO application has been inserted and connected with the rest of the model, enabling the application to have access to movement data and to add new graphical elements to the existing scene.

Several external parameters can be specified to customize the behavior of the system. ServerAddress and serverPort specify the IP address and port number of the computer, on which the server thread is running. Posx, posy and posz values define the X, Y and Z coordinates of the initial user position, while rot gives the current rotation around the Y-axis. MinDist and maxDist are used to determine the sound attenuation limits, as explained in section 4.1.1. If certain parameters are not specified, the default values are used instead.

In order to connect the VIRACO users, a server program must be started on a machine with known Internet address prior to starting any client programs. A few options can be selected at startup (Fig.6), then the program can be minimized and run in the background.

5 Conclusion
The VIRACO package offers quick and easy upgrade of stand-alone VRML worlds into multi-user networked virtual environments. Real-time audio communication and representation of users with avatars help to create an immersive feeling of participating in a collective happening. Platform independence makes it possible to run the system on a variety of hardware and software combinations.

The application is particularly suitable for creation of various virtual meeting places, where users can meet and chat. For example, fans from all around the world could connect to a certain web site, where a famous music performer would hold a virtual press conference. They could listen to the musicians interview, see his new image, ask him questions and share their comments with other visitors. Another use can be found in education, where VIRACO could connect users from different geographical locations into a single virtual classroom.

The present VIRACO application is limited to multicast network use, because multicasting improves the efficiency of the system, but also prevents its usage over the Internet. In order to enable the Internet usage of VIRACO, the package will be reshaped and unicasting will be used.

Because of its open and object-oriented structure the VIDER0 system is easy to upgrade. In the future, the performance optimization and the addition of a certain level of intelligence are planned. For example, if two users are far away from each other, they dont see nor hear the other one. That means there is no need to transmit the audio and positional information between the two clients and so some resources can be saved. Another big functional improvement would be the addition of support for shared objects and events, although it would complicate the building of environments. And users would probably prefer to use their own avatars for presentation in VE, so an option to import custom representation models would be welcome.

On the other hand, a new standard VRML 200X is being prepared by the X3D Task Group of the Web3D Consortium [11]. It will include support for multiple users, streaming of media and world data, additional graphical elements, support for XML and many other features. With this functionality, VRML will become more powerful for usage in interactive multi-user environments. Anyway, there will be still enough space for further improvements, like adding haptic/kinaesthetic features, which would enrich the virtual environments one big step forth.

References:


