A SIP SPATIAL AUDIO SERVER FOR THE EVE PLATFORM

Ch. Bouras
Research Academic Computer Technology Institute (CTI), Greece and
Computer Engineering and Informatics Department (CEID), University of Patras, Greece

V. Triglianos
Computer Engineering and Informatics Departmet (CEID) University of Patras, Greece

Th. Tsiatsos
Department of Informatics, Aristotle University of Thessaloniki, and Research Academic
Computer Technology Institute (CTI), Greece

Keywords: Virtual reality, Multimedia Collaboration, SIP, Spatial 3D audio, Networked Virtual Environments, Multimedia systems, architecture, and applications.

Abstract: When it comes to 3D Virtual Environments it is well known that 3D sound is of great importance to the whole interactive experience. The percept of sound is a major counterpart for the eyesight, since it describes the location, the momentum and the attitude towards the listener, of each surrounding entity. The sound can offer precision of spatial perception that cannot be achieved by the eye itself. Thus the support of 3D surround sound of high fidelity is mandatory for a 3D Virtual platform. The evolution of Internet telephony led to the introduction of new session establishment and management protocols. The most important of them, the Session Initiation Protocol (SIP), is a robust, lightweight reliable and fast application-layer control (signaling) protocol that is highly adopted for creating, modifying and terminating sessions. This protocol can be of extreme importance in establishing audio sessions for multi-user platforms. This paper presents the work done for developing a SIP 3D spatial audio server for a multi-user virtual environments platform, called EVE, in order to support 3D spatial audio.

1 INTRODUCTION

The increment of the available network bandwidth has enabled the development of richer web applications. In the field of Networked Virtual Environments this is reflected in the deployment of as realistic as possible virtual worlds. Virtual 3D environments offer a realistic abstraction of reality in an optical fashion. However, 3D sound is of equal importance in order for an interactive 3d environment to be realistic A key point to the realism is the sound capabilities it provides. Plain audio support is an important feature; however the best results are achieved with the addition of 3D spatial sound.

Audio support in virtual platforms practically means the digitalization of speech. Speech, one of the most effective means of communication that humans possess, is a time-efficient, emotion-declaring way to communicate. Thus, the introduction of speech is virtual platforms is time saving, since no message writing is required, while at the same time the voice contributes to more realistic communication and interaction between the users.

3D spatial audio contributes to a best perception of the environmental entities, especially when the user has no eye contact with them. The user by hearing 3D sound obtains information about the 3D location, and the direction that the entity that emits the sound moves. Moreover, depending on the intensity and the tone of the sound, a user can be aware, to some extent, of the intensions of the entity towards the user as well as the psychological

Bouras C., Triglianos V. and Tsiatsos T. (2008).
A SIP SPATIAL AUDIO SERVER FOR THE EVE PLATFORM.
In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pages 395-402
DOI: 10.5220/0001095203950402

composition of the entity. These psychological effects that arise from the 3D spatial sound combined with the perception of space, lead to a very realistic interaction in two fashions: between the users, and between a use and the virtual space.

Bearing this in mind the authors of this paper aims to enhance such a platform called EVE (http://ouranos.ceid.upatras.gr/vr/) which is a networked virtual environments platform that supports the following characteristics:

- Flexibility and good rendering quality, since the large set of all X3D nodes can be used to create worlds that are visually more appealing, as well as better defined compared to a VRML world.
- A consistent shared 3d virtual environment for all users connected to the platform, which is highly interactive and allows for all possible functionality of an X3D world to be reliably shared among all connected clients. In other words EVE is stable in terms of network and multi-user behavior.
- An efficient physics system functioning locally on each client's machine, which is provided by the Xj3D library and based on the ODE open-source physics engine, as well as an efficient sound system.
- Text chat and audio communication, using H.323 for audio and chat bubbles for text chat.
- User roles and user management.
- Support for avatar gestures and body language
- A flexible, fully customizable and open client-multiserver architecture.

EVE platform featured spatial H.323 sound since it was introduced. In order to increase performance, and stability as well as to emphasize the distributed nature of the platform a new audio server, utilizing the latest most popular session protocols and codec technologies and at the same time featuring a new algorithmic approach, was introduced. The new server is based on the SIP session protocol to establish multicast sessions as well as using RTP for audio data transmission. The spatial effects are implemented by the X3D Sound Node interface.

A fair amount of work has be done in on spatial 3D sound. The majority of today's 3D games, single or multi-user, feature spatial 3D sound. However the sound that is used in this type of applications is pre-recorded. When it comes to CVE's where live streaming sound needs to be converted to spatial 3D sound, little work has been done. Good examples are the work in Liesenborgs (1998) and Macedonia et al (1995).

The work described in Macedonia et al (1995), is based on multicast networks. We want to avoid this solution due to the fact that multicasting is not available in every network. Furthermore, our work differs from Liesenborgs (1998) in terms of scalability, complexity of the platform that is embedded into, and technology. Our scope is to utilize the latest technologies into our platform, while using Java API's in order to maintain the well-known and commonly accepted benefits of Java.

When it comes to spatial distributed sound there are two key subjects to address. The first is the protocol the algorithms and the codecs that will be used to establish sessions and reproduce sound between users. The second is the algorithms that will create the illusion of 3D sound.

As far as the protocol is concerned, the popular Session Initiation Protocol – SIP was used. The transmission of the audio data utilizes the RTP protocol while the task of spatializing the sound is accomplished by the X3D Sound Node.

X3D offers the Sound Node interface. This node features built-in spatialization and attenuation audio algorithms. This solution is both fast and simpler to develop since no external solution is used more.

The paper is structured as follows: The next section demonstrates the architecture of the proposed solutions. Following this (in paragraph 3) the mechanism and the algorithms that form the selected solution are described. After that (in paragraph 4) the implementation issues are discussed while on section 5 some performance tests are presented. In Section 6 some concluding remarks and planned next environment platforms either commercial products or steps are briefly described.

2 EVE SPATIALIZED AUDIO ARCHITECTURE

The EVE platform is based on a client-multiserver architecture, which allows a simple sharing of the computational load among multiple servers. The main servers used by the platform are the connection server, the VRML server (or data server) and a series of application servers, which add specific functionality such as audio and text chat to the platform. The architecture of the EVE platform is displayed in Figure 1.

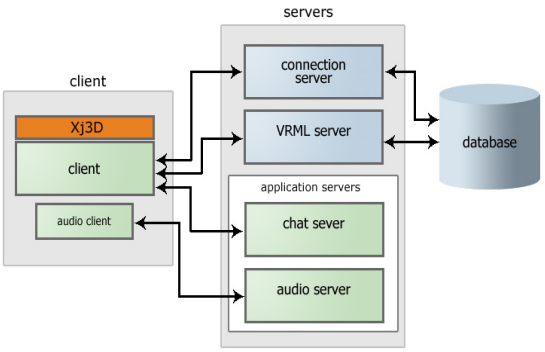

Figure 1: Architecture of EVE.

Due to the multi-user and client-server characteristics of the platform, not spatialization of audio but also networking and communication are important parts of the whole process. Moreover, the real time nature of the process requires bandwidth saving and fast audio processing to deliver continuous playback of the sound. In order to provide an architecture that would balance the bandwidth, complexity and processing costs many candidate scenarios were examined. The best two are presented in this section. The two architectures are different in the way that the audio is processed in order to become spatialized. The procedure of connecting to the platform until the spatial audio is produces consists of four discrete stages, from the client's point of view:

- The establishment of the connection to the SIP Spatial Audio server
- The capture of the audio data
- The transmission of the audio data
- The process of spatializing the audio data
- The reception of the audio

For the common part of establishing a connection to the SIP Spatial Audio server, capturing the audio data and transmitting the audio data, follows a description of the technologies that are implemented.

The *Session Initiation Protocol* – SIP is ideal for the session establishment. This lightweight, transport independent protocol has proven to be very reliable and robust thus making it the most popular session protocol nowadays.

Real Time Protocol - RTP a very popular protocol for real time data, including features like *Real-time Transport Control Protocol* - RTCP a protocol that provides control information for RTP flows, is chosen for audio data transmission.

As far as the audio capture is concerned, the *Java Media Framework API*, which provides convenient classes and methods for media manipulation is used.

In the following two subsections, the above mentioned stages are described for each of the suggested solutions.

2.1 Solution 1: Audio Spatialization using X3D Nodes

In this solution the spatialization of the audio is performed by an X3D Sound node (Figure 2). The client's side architecture is examined first. The establishment of the connection is accomplished via the SIP protocol. The client's applet makes a call to the SIP Spatial Audio Server and a server port is reserved for the connection with the client. After the session has initiated, an RTP stream, using the JMF API, is established with the server, in order for the audio data to be transmitted. At the same time the client's capture device captures audio data, again utilizing JMF API's classes, and transmits them through the RTP stream. The X3D browser of the applet receives the audio data encapsulated, by the SIP Spatial Audio Server, in X3D AudioClip nodes. The playback of the audio is performed by X3D Sound nodes that use the AudioClip nodes as their sources.

The X3D Sound node features built-in spatialization and attenuation audio algorithms. This solution is both fast and simpler to be developed since no external solution is used.

Regarding the server side, the following procedure takes place. The server is waiting for new SIP calls on a dedicated port. After an incoming call from a client is accepted, a new port is assigned for the communication between a server thread, dedicated in servicing the specific client, and the client. This thread establishes an RTP stream with the client for receiving audio data. Concurrently, the audio server acquires information of the user's avatar location and orientation in the virtual world through the VRML Server. This information will be used to reproduce the sound like it is being emitted from the avatar's mouth. For each user an X3D AudioClip node and an X3D Sound node are instantiated via the xj3d API and are added in the graph scene of the virtual world. A file is created to which the audio data are continuously appended to. The AudioClip node's url field is given that file as a value, while the Sound node's fields direction, location and source fields are given the values of the

avatar's mouth direction, avatar's mouth location and the AudioClip, respectively. The key point in this solution is the entrusting of the audio spatialization to the X3D Sound node. This node can produce spatialized audio by setting appropriate values to the specialized fields.

When the two new X3D nodes are added to the scene the VRML server sends them to the client where the client's X3D browser starts immediate playback of the Sound node.

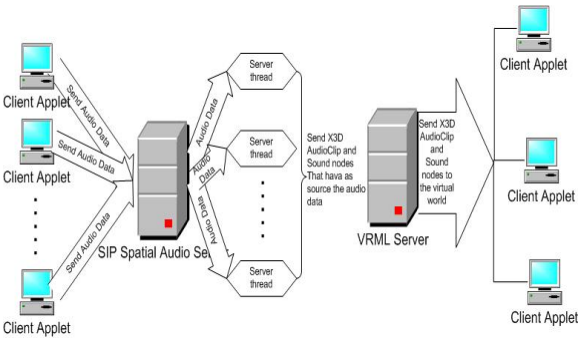

Figure 2: Audio spatialization using X3D nodes.

2.2 Solution 2: Audio Spatialization Exploiting a Spatialization Algorithm

The second solution (Figure 3) uses a spatialization algorithm in order to spatialize the audio. The process of session initiation, audio data capture and audio data transmission is the same with the one described in the first solution. What is different in that solution is the way that the SIP Spatial Audio Server processes the audio in order to make it spatialized. That, reflects also to the way that the client receives and playbacks the processed audio. The next paragraphs illustrate the differences in both the client and the server's side.

The major client's side change is the way that the audio is received and being produced. The user receives a mixed RTP stream, that is, all the streams of client audio data to be heard, have been processed from the SIP Spatial Audio Server and have been mixed in to one stream. Then it reproduces the sound via a JMF Player.

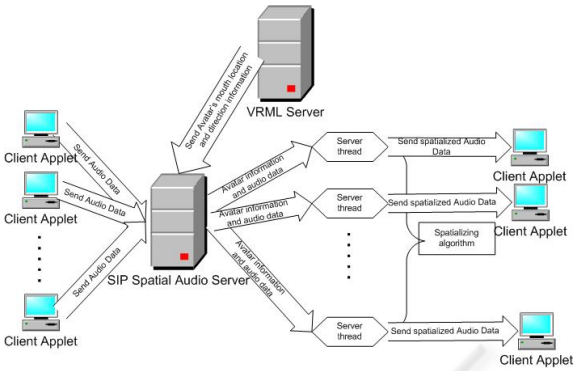

Figure 3: Audio spatialization exploiting a spatialization algorithm.

The most important changes are in the server's side. When the server receives the streams from the client together with the avatar information mentioned in solution one, it uses a spatialization algorithm for each stream. This algorithm is invoked from each server thread that serves a client. This algorithm given a stream and an avatar information, produces spatialized audio that sounds like it is being emitted from the avatars mouth. The server thread invokes that algorithm as many times as the online users connected to the SIP Spatial Audio Server, Each invocation of the algorithm instantiates a new thread that executes the algorithm in order to continuously process each stream. The processed streams are mixed by the server thread into one stream and are sent back to the client.

2.3 Discussion

In this section the paper presents a comparison between the two solutions and the selection of the most appropriate one. The two proposed solutions will be examined in terms of bandwidth and processing costs, complexity, as well as EVE depended issues.

The multimedia nature of the platform's content disambiguated in high bandwidth consumption. The addition of an extra multimedia factor (in this case the audio) must be carefully implemented with as low bandwidth cost as possible. Regarding the client, the first solution requires one extra stream for transmission-only purposes, after the client - server handshaking has been achieved. The reception of the audio is accomplished by the already established udp stream of the VRML Server. As result the network side of the platform is lightly overloaded. On the other side, the second solution demands both

an inbound and an outgoing stream from the client's side, thus being more costly in bandwidth.

The processing cost, especially for the client is a very important factor for deciding which solution will be implemented. The combination of java with X3D content and the fact that the platform runs via an X3D-enabled HTML browser lead to a relatively high processing cost. The addition of spatial audio support must, therefore, be as light, in processing cost, as possible. The server processing cost is very important as well due to the number of clients a server may be requested to server. The first solution adds very little processing cost to the client, since no extra modules are required to playback the audio. The only extra cost is that of the X3D browser reproducing the audio which is comparatively small comparing it with the need for a new module. The second solution is light as well for the client since only one JMF Player class instance is required for audio playback.

However, that is not the case for the server's side. The invocation of n^2 threads, if n is the number of clients, demands much more processing time than the n threads required by the first's solution server. In addition the spatialization of the audio is performed exclusively in the server, when the second solution is used while on the first scenario each client's applet is responsible for the process. The cost of spatializing the audio becomes a vast process cost when the clients increase greatly in number. Conclusively the first solution comes with lower processing cost than the second one.

In terms of complexity the picture remains the same. The first scenario relies for the spatialization to the internal operations of the X3D browser, while the second invokes n^2 times the spatialization algorithm, while at the same time the SIP Spatial Audio Server is assigned with the complex task to mix n streams for each user, where n is the number of online clients.

To conclude with, the first solution is in all of its aspects better than the second. However, there is a platform dependent issue that need careful examination in order to avoid a very uncomfortable situation. Due to that fact that the first scenario relies to the continuous communication between the VRML Server and the SIP Audio Server, a breakdown of the SIP Audio Server could lead to a VRML Server exception and vice versa. Nevertheless given the necessary attention this issue can be sustained, and as result, given the overall dominance of the first solution, the first scenario was chosen.

3 DESCRIPTION OF SIP SPATIAL AUDIO MECHANISM

In this section the mechanism behind the SIP Audio Server is described. The mechanism consists of three main components. The SIP component, the capture component the RTP component and the spatialization component The following three paragraphs describe each one of the above respectively.

Each EVE client applet features an integrated SIP client (Figure 4). When the EVE Applet connects to the connection Server of the platform, a unique port is granted for SIP use. The applet passes this port parameter to the SIP client, which sends an SIP INVITE message to the SIP server in the previously mentioned port. Subsequently, the client waits for the SIP OK message. As long as the server accepts the invitation, a server thread is created to serve the client. The thread establishes an rtp receive stream with the client while the client establishes an RTP send stream with the server thread. When the client decides to disconnect from the platform a SIP BYE message is sent to the server. When the client receives a SIP OK message from the server, the session ends.

Once the SIP session is established the client's applet invokes the methods for capturing the sound. Firstly, a list of the available capture devices is examined until an appropriate for sending audio data, is found. Next, follow the instantiation of a processor that receives the capture data and produces a data source in the specified format that is continuously filled with captured audio data. This data source is used by the RTP stream to send the audio data to the sever.

The RTP manager creates an RTP send stream and passes to it as argument the data source that is produced by the processor. Once this is accomplished the stream starts sending the audio data of the audio source. On the server side, the server thread that corresponds to the particular client instantiate an RTP manager, which, in turn, creates a receive stream that stores the received data to a buffer file for a constant amount of time. When this amount time has elapsed a second buffer is being written for the same amount of time while the first is flushed. This procedure is continuously repeated with one buffer being filled with data and the other being flushed.

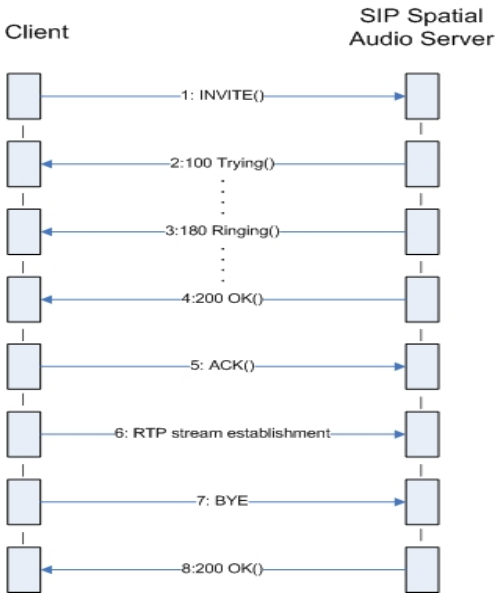

Figure 4: SIP session.

Simultaneously with the capture setup, the client's applet instantiates an RTP manager that will manage the rtp session (Figure 5).

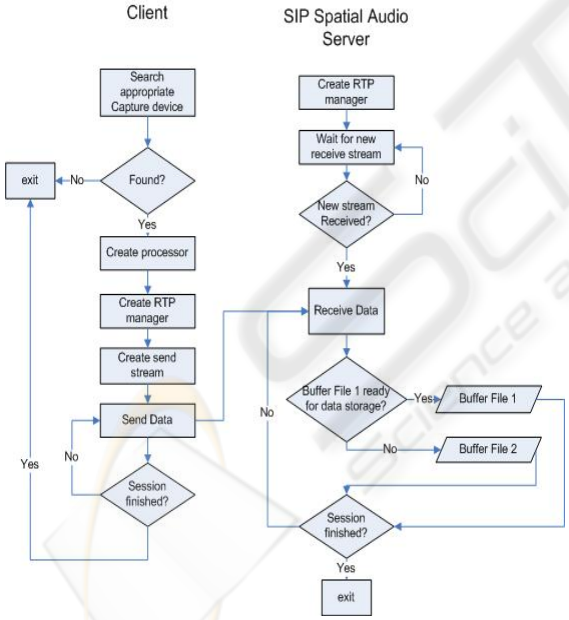

Figure 5: RTP streaming flowchart.

The server thread when the RTP stream is established, constantly, acquires information from the VRML server about the client's avatar. Then two X3D AudioClip nodes and two X3D Sound nodes are created. The Sound nodes' fields of location and direction match those of the avatars mouth.

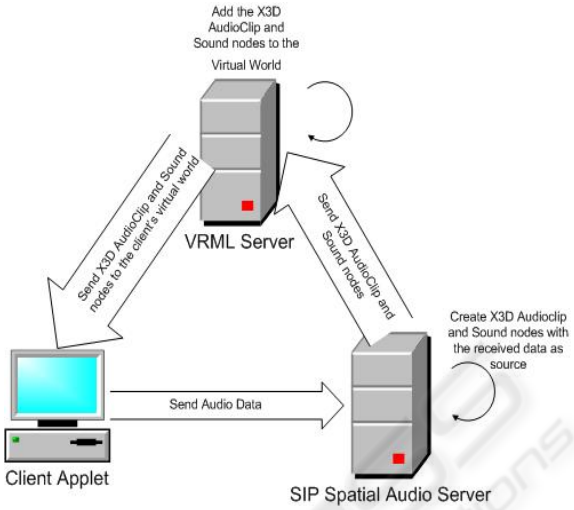

Figure 6: Spatialization process.

Each Sound node has one, different with one another, of the AudioClip nodes as its source field, while each AudioClip node has as URL field one of the two buffer files (Figure 6). Once the first buffer file is filled with data the corresponding AudioClip starts playback and the corresponding Sound node produces the effect of spatialization. When the first AudioClip finishes the second set of nodes starts playback. This procedure is repeated as far as the user is connected. It must be stated that the actual X3D nodes are created on the client-side. However since the AudioClip takes URLs as a source the client has access to the buffer files that are stored on the SIP Audio Server.

4 IMPLEMENTATION ISSUES

In this section the implementation issues are discussed. The main technologies, classes and methods of the SIP Spatial Audio Server implementation are presented.

4.1 SIP Implementation

Because of the fact that the EVE platform is Java – xj3d based, an java implementation of the protocol was needed, to maintain its non-commercial, cross platform characteristics. JAIN – SIP was chosen as an API, nist-sip 1.2 was used as reference implementation. These are the main classes used:

- SipStack
- SipFactory
- SipProvider

- ClientTransaction
- MessageFactory

Three are the main methods invoked:

- processRequest (RequestEvent). A custom method for processing SIP requests such as INVITE.
- MessageFactory. createRequest. A method for creating SIP requests such as INVITE

4.2 RTP Streaming and Audio Capture

The RTP streaming and the audio captured management tasks, is performed by custom code tha utilizes the *Java Media Framework API* – JMF.

Five are the main classes used for the audio capture:

- javax.media.CaptureDeviceInfo
- javax.media.Processor
- javax.media.MediaLocator
- javax.media.Manager
- javax.media.format.AudioFormat
- javax.media.protocol.DataSource

A list of the most important methods for the audio capture:

- Vector CaptureDeviceManager.*getDeviceList*
- DataSource Manager.*createDataSource(MediaLocator ml)*
- Processor Manager.*createProcessor*(DataSource ds);
- Javax.media.control.FormatControl.*setFormat*(AudioFormat af).
- DataSource Processor.getDataOutput()

The capture format that is used in this implementation is in linear encoding, has 8000 Hz sample rate, 8 bits of sample size and is monophonic. The streaming format was of the same characteristics. We used this relatively low quality settings in order to save bandwidth and processing resources.

Main classes used for the RTP streaming are the following:

- javax.media.rtp.RTPManager
- javax.media.rtp.SendStream
- javax.media.rtp.ReceiveStream
- javax.media.protocol.PushBufferDataSource
- lavax.media.protocol.PushBufferStream
- javax.media.rtp.event.NewReceiveStreamEvent

A list of the most important methods for the RTP streaming is the following:

- RTPManager.NewInstance()
- RTPManager.createSendStream (DataSource dataSource, int streamIndex)
- NewReceiveStreamEvent.getReceiveEvent()

4.3 Audio Spatialization

The spatialization of audio is performed by the X3D Sound Node. The Sound node specifies fields that affect the spatialization of the sound. The sound is located at a point in the local coordinate system and it is emitted in an elliptical pattern. The location is specified by the *location* field, while the direction vector of the ellipsoids is specified by the *direction* field. There are fields that specify the maximum and minimum values to where the sound is audible, that is the maximum and minimum lengths of the two ellipsoids along the direction vector.

A very crucial field is the *spatialize* field. If set to *TRUE* the sound is perceived as being directionally located relative to the viewer. If the viewer is located between the transformed inner and outer ellipsoids, the viewer's direction and the relative location of the Sound node is taken into account during playback. In our implementation this field is set to *TRUE*, resulting in a very realistic spatialized audio playback.

The sound source specified by the field source is an AudioClip node. The AudioClip node specifies an url field, that in our implementation is the url of the buffer file, which is used as source. In order to change between the two sets of nodes, we used an ecma script which sets the *startime* field of one AudioClip that waits to start, equal to the *stoptime* field of the currently playing AudioClip.

5 CONCLUSIONS - FUTURE WORK

This paper presents the addition of an SIP Spatial Audio Server to the EVE platform. This new server enhances to a great extend the realism of the virtual world and expands the interaction capabilities of the platform. In addition it utilizes the latest technologies in this field which makes EVE an equivalent alternative to commercial solutions.

Our next step is to simulate the two proposed solutions in order to conclude to experimental results of each solution’s efficiency and stability.

REFERENCES

Liesenborgs, J., 1998. Voice over IP in networked virtual environments, Computer Society Press.

Macedonia, M., Brutzmann, D., Zyda, M., Pratt, D., Barham, P., Falby, J., Locke, J., 1995. NPSNET: A multi-player 3D virtual environment over the internet. In Pat Hanrahan and Jim Winget, editors, 1995 *Symposium on Interactive 3D Graphics,* pages 93-94. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

Bouras, C., Panagopoulos, A., Tsiatsos, T., "Advances in X3D multi - user virtual environments", IEEE International Symposium on Multimedia (ISM 2005), Irvine, California, USA,, 12 - 14 December 2005, pp. 136 – 142.

Bouras, C., Giannaka, E., Panagopoulos, A., Tsiatsos, T., "A Platform for Virtual Collaboration Spaces and Educational Communities: The case of EVE." Multimedia Systems Journal, Special Issue on Multimedia System Technologies for Educational Tools, Springer Verlang, Vol. 11, No. 3, pp. 290 – 303, 2006.

Bouras, C., Tegos, C., Triglianos, V., Tsiatsos, T., "X3D multi-user virtual environment platform for collaborative spatial design". The 9th International Workshop on Multimedia Network Systems and Applications (MNSA-2007), Toronto, Canada,, 25 - 29 June 2007, (to appear).