Adapting multiuser 3D virtual environments to heterogeneous devices

Regina Borges de Araujo[1], Alessandro Rodrigues e Silva[1] and Glauco Todesco[2]

[1]Departamento de Computação - Universidade Federal de São Carlos (UFSCAR)
Caixa Postal 676 - 13.565-905 - São Carlos - Sp - Brasil
{regina, arsilva}@dc.ufscar.br

[2]LSI - Laboratório de Sistemas Integráveis EPUSP - USP
todesco@lsi.usp.br

Abstract

With the growing dissemination and reliability of wireless networks and the emergence of devices with increasing processing and communication power, applications that up to now were restricted to the PCs are being envisaged to run on devices as heterogeneous as wrist clocks, refrigerators with access to the internet, mobile phones, PDAs, set-top-boxes, game consoles etc. Application development for this myriad of devices and networks with different capabilities requires special attention from the software programmers and designers - especially when these applications are shared among multiple users. Application adaptation, which allows a software to react to device and environment resource variations, is an important process to fit the application to a certain device configuration. A large amount of work has focused on the adaptation of multimedia such as text, images, audio and video. Less attention has been given to 3D media adaptation - firstly because of the complexity involved in the 3D application adaptation, and also because true marketing opportunities for 3D applications in heterogeneous devices have just began to emerge. This paper analyses 3D media adaptation as a nonfunctional requirement for 3D multiuser virtual environment applications. An adaptation framework is proposed that can be integrated to the MPEG-4 standard to offer a solution to the adaptation of 3D multiuser virtual environment applications, which can be accessed from heterogeneous devices with different capabilities. The advantages of integrating the framework to the MPEG-4 standard are twofold: it favors the creation of complex applications, with high degree of interaction, such as multiuser 3D collaborative environments; and makes easier to build these applications for heterogeneous devices (from cellular phones to PDAs and set-top-boxes) since MPEG-4 is aimed at small mobile devices and narrowband networks, such as some wireless networks.

Keywords: multiuser virtual environments,3D media adaptation, MPEG-4.

1. Introduction

Multiuser 3D virtual environments – 3DVEs are characterized as multidimensional virtual tri-dimensional environments, which can be shared by multiple participant users who interact with the application environments ranging from collaborative projects to multiplayer games, military and industrial training, etc. The environment is inhabited by dynamic entities, which can be driven by users (through the avatar – the user´s geometric representation on the VE) and by simulation (scripts or Artificial Intelligence). The environment contains also static entities such as buildings, trees etc., that can be subject to modifications during the application. When a user triggers an action, this action has to be reflected locally as well as to all remote users´ client terminal (device used by the user to access the VE). All this must happen within an acceptable latency so that all users share the same view of the VE and can interact with the system. When users use heterogeneous devices to exchange data among themselves, one more challenge emerges, which is to adapt the application to fit the device and network capabilities. Until recently, popular 3D VE applications, such as games, were seen basically in personal computers. This changed dramatically with the growing dissemination and reliability of wireless networks and the emergence of devices with increasing processing and communication power. Mobile 3D and other more complex applications pose many challenges, such as high costs of wireless connections, high latency and frequent connection disruptions. Moreover, one of the mobile computing application goals is to have the application to "follow" the user; however, it is a complex task to provide, in a transparent way and without user aid, adjusts to eventual changes in the environment. This paper handles

Adaptability, an important requirement for 3D VEs, which involves multiple users accessing a shared VE from heterogeneous devices and networks with different capabilities. A framework is described that supports data and control adaptation of multiuser 3D virtual environments. In order to provide a wider scope of application, MPEG-4 was chosen as the 3D VE supporting technology. The proposed framework is integrated to the MPEG-4 standard, through the MPEG-J and MPEG-4 multi-user extensions, to offer an MPEG-4 conformant solution for adaptation of 3DVEs to heterogeneous devices. The advantages of integrating the framework to the MPEG-J and MPEG-4 are twofold: 1) it favors the creation of complex applications, with high degree of interaction, such as multiuser 3D games, training, etc.; and 2) it makes easier to build these applications for heterogeneous devices (from cellular phones to PDAs and set-top-boxes), since MPEG-4 is aimed at small mobile devices and network narrowband, such as some wireless networks.

The rest of the paper is organized as follows: Related work is discussed in section 2. Section 3 presents 3D VE supporting technologies available. Section 4 describes the *3DVEAdapt,* a framework for 3DVE application adaptation. An evaluation of the framework, through a 3D multiuser game, is described in section 5, followed by conclusions and references.

2. Related Work

Most adaptation mechanisms or solutions found in the literature are concerned with audio and video adaptation. Less attention has been given to 3D virtual environment adaptation. Examples of audio and video adaptation include [6] and [3]. Some adaptation mechanisms are application independent, such as the framework by Chang e Karamcheti [5]. These mechanisms provide continuous adaptation under changes in resources capabilities through two components: *Tunability Interface* by which the user may configure the application; and a *Virtual Execution Enviroment*, which emulates the application runtime under different situations of resource availability. Together, these mechanisms take decisions on when to adapt (continuous monitoring of the device capabilities and application progress) and how to adapt (according to user preferences). Not all resources are monitored though, only CPU and network bandwidth capabilities. Moreover, control adaptation is not dealt with, only data adaptation. ARTE - Adaptive Rendering and Transmission Environment for 3D Graphics [18], is one of the few 3D adaptation systems found in the literature that delivers 3D models in heterogeneous environments (devices and network). ARTE relies on four components: a monitor to control changes in the device and network capabilities; different hierarchical organizations to represent different 3D models; an adaptive selection mechanism that considers environment, model structure and user preferences that determine the most apropriate modality for each of the 3D model that best fit a given resource configuration; and a transcoding engine that basically converts 3D data to the modalities selected by the adaptation mechanisms. ARTE considers the application resources (such as frame rate) but not the device capabilities (e.g. CPU and memory) what can impact on the rendering performance and quality. Heterogeneity in Networked Virtual Environments is handled in [30]. In this system, a client-server architecture, Switchboard Architecture (SA), is described that map individual users' preferences and policies to simple linear equations and inequalities that form the mathematical model of the functions of the system. Although it is a novel idea and can work for simple VEs, it remains to be seen if complex VEs could be easily expressed as math models. Existing frameworks, such as QUICK [4], manages display and representation of objects according to the proximity or interest of the user (level of detail), but it is not a complete structure to provide adaptation according to resource variations. Other works are based on the exchange of video and/or audio in a networked VR [8]. Other mechanisms that contribute to filter data and so minimize network bandwidth and device resources have been widely discussed in the literature. In [1], a review of several mechanisms is carried out.

In this paper, a framework to adapt 3D VEs is described that handles user preferences, devices and network resource variation by providing control and data adaptation. This framework is integrated to the MPEG-4 standard. Next section discusses available 3D VE supporting technologies and the choice of MPEG-4 as the mechanism to which the adaptation framework is integrated to.

3. 3D VE Supporting Technologies Available

There are many available technologies to build 3D virtual environments, especially for the WEB. Some of these technologies, such as ShockWave3D [13], QuickTime VR [27], Pulse 3D [25], Adobe Atmosphere [11], Demicron [12], and others, are not open source or public domain and so, were not explored in this paper. A review of these technologies can be found in [10]. Open standard technologies like VRML, X3D, Java3D and MPEG-4 are summarized below.

VRML

VRML was devised initially for the development of rich virtual environments in the WWW [19]. Cris Marrin [14] identified several problems arisen from the complexity

of the VRML specification. These limitations result in the generation of complex and unstable VRML browsers what reflects directly on the performance and presentation quality of the scene as well as the integration to multimedia. For example, in a mobile 3D virtual environment application, a trade-off can be made between quality of presentation and rendering rate what is not supported by VRML. Moreover, VRML does not have a binary format what causes high latency as usually the VRML file format is large and can take a long time to load.

X3D

X3D [17] is an Open Standard XML-enabled for 3D content delivery. It is not a programming API nor just a file format. An X3D file combines geometry and runtime behavioural descriptions and is considered as the next generation of the VRML 97. Changes to X3D overcomes the main VRML limitations - those are summarized as follows [17]: Expanded scene graph capabilities; Revised and unified application programming interface; Multiple file encodings, which describes the same abstract model including XML; Modular architecture; and Expanded Specification Structure. Open software references available include Xj3D [18], which defines a toolkit for VRML97 and X3D content written completely in Java.

JAVA 3D

Java 3D [29], a scene graph based 3D API, was primarily provided by Sun Microsystems but is now a community open source project. Java 3D runs on top of OpenGL or DiretcX - its API enables the creation of 3D contents. The main advantages of Java 3D is the rich sets of API available in all Java distributions. Another characteristic of Java3D is the loader classes - a loader class can read a 3D scene file (not a Java file) and create Java 3D representations of the file's contents that can be selectively added to a Java 3D world or extended by other Java 3D code. There are 14 public loaders available (including for 3D-Studio, AutoCAD, VTK and VRML). It is also possible to write custom loaders for Java 3D. Its main drawbacks are its long learning curve, and the need to have a virtual machine in the devices, which can have limited resources.

MPEG-4

MPEG-4 is an ISO/IEC (ISO/IEC JTC1/SC29/WG11) standard, developed by the MPEG *(Moving Picture Experts Group)* for coding and delivery of different media formats in a wide variety of networks and computational plataforms [23]. Differently from VRML, MPEG-4 deals with the composition, rendering, compression, synchronisation and distribution of multimedia objects.There are currently some versions of the MPEG-4 Player (MPEG-4 client terminal) available for interactive TV set-top-boxes. The deployment of the MPEG-4 standard in cellular telephones and mobile devices in general is already on course and it is just a matter of time before we can see complex multimedia applications in these devices, through the MPEG-4 Player. An extension to the MPEG-4, which supports multiuser 3D VEs, MPEG-4MU, is widely based on the ***Pilot/Drone*** mechanism proposed by the ***Living Worlds*** specification [16]. According to that specification, ***Pilots*** are master copies of shared objects in a VE. Any changes in the *Pilot* state or behavior are replicated to other instances of that object, which are named ***Drones*** (***Pilots*** replicas). The replication of changes in the *Pilot,* or any other object in the graphics scene, can be realized basically through the BIFS-Command Protocol. When these commands arrive at the remote users´ client terminals, they are locally processed, causing the update of the scene, so that all users that receive a BIFS command have a consistent view of the shared VE. The MPEG-4MU architecture components responsible for managing multi-user sessions and scene synchronization are the MUTech Session Controller (MSC) and the MUTech Bookeeper (MBK) components, respectively [32]. The MPEG-4MU defines a standard interface for the access to these components and suggest the set of functions these components should perform - it is then left to developers how these functions are implemented. The integration with XML is a major target of MPEG-4, because it would enable authors to integrate content from different providers into one scene. MPEG-4 supports the XML integration through the Extensible MPEG-4 Textual Format (XMT). The XMT format is the use of a textual syntax to represent MPEG-4 3-D scene descriptions like X3D. XMT was designed to provide content authors with the ability to exchange their content with other authors while preserving their intentions in the text format. XMT provides interoperability between MPEG-4, Extensible 3D (X3D), and Synchronized Multimedia Integration Language (SMIL). The XMT format can be interchanged between SMIL, VRML, and MPEG-4 players. The componentization is another important way to create content suitable for different platforms and MEPG-4 BIFS is one of the supported components. VRML limitations can be better overcome by the MPEG-4 framework, which is a powerful platform to support 3D graphics along with natural audiovisual data. Moreover, X3D can be supported in an MPEG-4 scene through the MPEG-4 AFX [15].

The MPEG-J Extension to the MPEG-4 standard

MPEG-J is a programmatic system, an extension to the MPEG-4 standard, which supports complex scene control within MPEG-4 content, through Java classes.

These classes are received by the MPEG-4 Player as MPEG-lets [20]. MPEG-J supports the building of complex 3D multimedia presentations, where the scene can be manipulated directly by the application, through the MPEG-J APIs [20]. The APIs can also provide the reading of static and dynamic resources from devices. For instance, the application can gracefully degrade an image against resource fluctuations, such as the CPU load. The problem with these APIs is their limitation to monitor resources as a complete solution, as is the case of 3D VE application adaptation. Next section describes a broader solution, through an adaptation framework integrated to the MPEG-4 player. The solution extends the actual MPEG-J definition for a better and more complete control of adaptation.

4. A FRAMEWORK TO SUPPORT 3D MEDIA ADAPTATION INTEGRATED TO MPEG-4

An application can be submitted to **data** and/or **control** adaptation. Data adaptation implies on a transformation of the data often used by the application to versions that fit into available varying resources. This adaptation is made without losses of representation. For instance, a texture can be transformed into another texture version with smaller resolution. On the other hand, control adaptation means to modify the application control flow, i.e., the application behaviour. For instance, as soon as an application learns that network delay has had an increase, it could introduce a buffer for data reception, at runtime, to reduce latency [21].

Adaptation is driven by adaptation policies. The type of application, as well as the data type of the application usually dictate the adaptation policies. An adaptation model should take adaptation policy into consideration (what to adapt, when to adapt, how much to adapt, costs consideration etc), besides other issues such as detection of interest conflicts and how to sort them out, user intervention degree in the adaptation policy etc. [28].

In order to support high interactive 3DVEs with low latency across multiple plataforms, adaptation is employed in such a way that it is possible to fit the application according to device and network resources variation as well as behaviour of application (natural variation of frame per seconds - fps number). For the VEs to be adapted, it is necessary to identify the type of information processed and how it can be adapted. The information processed by the graphics subsystem include: Object and character models (geometric representation), terrains, textures, materials, movimentation and animation models (transformations on the models), effects and *cut-scenes* (small pre-rendered films). The audio processing uses sound and music files. Physical Modeling handles, through mathematical models, for instance, the behavior of objects that, within the VE, are subjected to physics laws, such as gravity, wave moviments, light refraction and reflection, among others. Artificial intelligence is responsible for the behaviour of the NPCs (Non-Player-Caracters) path-finding algorithms. This information is processed by engines of corresponding types: graphics, audio, tactile, physical modeling, AI engines – all with access to a common database [24] [2]. This approach is also used by Unreal Engine Technology [31] and RenderWare Platafform for building interactive multi-user 3D games. Next section describes a framework to adapt data and control in 3DVE applications.

4.1. *3DVEAdapt* - A FRAMEWORK TO ADAPT 3D VE APPLICATIONS

A framework was created that adapt 3D VEs application data as a result of monitoring fluctuations in the level of system and application resources. The framework, named *3DVEAdapt,* makes decisions whose goals are to keep the performance requested by the user. The framework can be used in MPEG-J applications to help programmers to insert adaptation mechanisms that react to the fluctuations in the level of resources of a device as well as to performance variations of the application itself (e.g., rendered fps rate variations due to differences in the scene complexity). The goal of the adaptation is to maintain the performance required by the 3DVE application user in heterogeneous devices. With *3DVEAdapt* framework, the programmer defines the resources to be monitored, as well as the mechanisms that will be activated in order to guarantee the performance level determined by the user. It helps programmers to insert adaptation mechanisms that deal with three issues: 1) device and network capabilities heterogeneity; 2) fluctuations of the user´s environment level of resources, resulting in loss of performance; and 3) variations of the scene complexity during 3D VE runtime, resulting in loss of performance. The adaptation mechanisms are identified dynamically and used on a per client terminal basis.

3DVEAdapt has the following components: Performance Specifier (PS), Content Adapter (CA) and Resources Monitor (RM), as shown in figure 1. Each component is represented by a running process. PS and RM are located at the client terminal and the CA is located at the same place MSC component is located (the MPEG-4 session control component described in section 3). PS is processed only once for each client terminal, whereas RM is ran while MPEG-J application is using it. CA remains active while the MSC component is available.

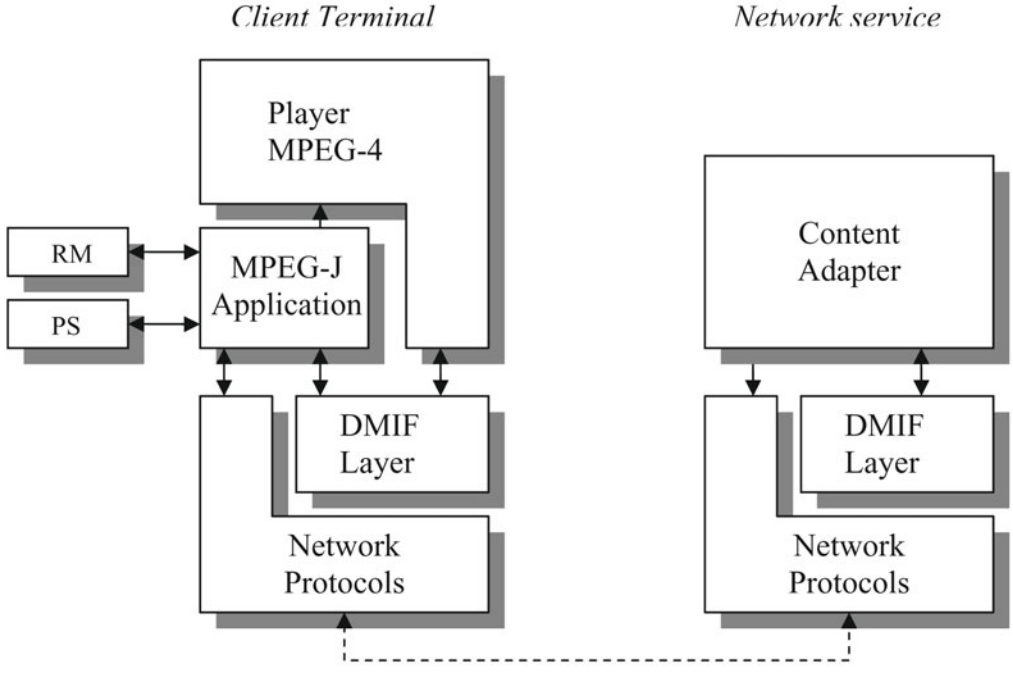

Figure 1: The *3DVEAdapt* framework components.

PS function is to determine client terminal capabilities when a multiuser 3DVE is running. The impact of the adaptation mechanisms, either present at the client terminal or at the CA, the handing out of the results to the programmer and the resources monitoring feedback in real-time are all responsibilities of the RM component. CA activates the adaptation methods before the data arrives at the client terminal. During the application processing, the adaptations made at the client terminal are triggered by the MPEG-J application. Next sections describe the policies that are followed by the framework components.

4.2. Adaptation policy: When to adapt ?

When the user downloads a 3DVE or a certain region of the 3DVE, adaptation is made before the data is transmitted to the client terminal in the following situations: PS checks that the client terminal does not support a specific data type; minimal fps required by the user cannot be achieved during the application runtime; the client terminal does not have enough network bandwidth for receiving and transmitting required data. The mechanisms are triggered by the MPEG-J application that receives feedback from the RM component when any resource reaches a threshold pre-established by the programmer.

4.3. Adaptation policy: Where to adapt ?

Depending on the situation, adaptation can be done at the client terminal or at the CA. An MPEG-J application can request the CA component for the adaptation of any data type, so long as there is support for such. This is also true for the adaptations made at the client terminal. Adaptations regarding network bandwidth and device capabilities are made at the CA, which is located at the same machine than the MSC component. Adaptations regarding all local resources variations, but network bandwidth and fps, are carried out at the client terminal.

4.4. Adaptation policy: What to adapt ?

Adaptation mechanisms were considered that trigger changes in the data and application control in order to reach better performance and flexibility based on the device capabilities and users´ preferences. The mechanisms that adapt application data are used in both the client terminal and the CA. These mechanisms can be divided in: 1) transmission media between users (voice and video messages) and transmission media between the client terminal and the Server (textures, audio, video and MPEG-4 files). The control adaptation mechanisms are used only at the client terminal to control the rendered fps rate. A small set was selected to be handled by the framework: number of objects rendered (decrease), Visual effects (decrease), physical modelling (remove), sounds played simultaneously (decrease), processing time to AI for the non-player-characters (decrease).

4.5. How to adapt?

Data adaptation in an MPEG-4 VE can be realized by changing the configuration and/or replacing the decoders associated to a certain data type. A data can be

decoded by using several configurations of the same decoder. Data sent from user to user can be recoded at the CA through a proper decoder. In case a decoder change occurs at the server side (CA), the same change has to be made at the client terminal decoder. For that, the client terminal must have support to the corresponding coder/decoder at the MPEG-4 Player. There are many ways to data adaptation in a 3DVE application. The presence of these mechanisms depends on their existence at the CA and at the client terminal, as well as the existence of the data type supported in the application. The *3DVEAdapt* framework provides methods that capture these mechanism capabilities and their impact on the client terminal through the PS. The framework is not concerned about the way a certain mechanism works. Rather, it is concerned with the impact these mechanisms cause on the VE. The implementation of a 3DVE varies from one another, so part of the tests realized by the PS have to be remade to all new 3DVE applications.

As occurs in data adaptation, control adaptation needs the required controls to be present in the application. Flow control adaptation means to change the logical flow of the application, e.g., to use or not physical modeling or dynamic shadows in a particular situation. For that, the application has to be written in a standard way, so that the adaptation framework can recognize the adaptation points. It follows below some examples of functions used for each type of data:

- ***Geometric 3D data:*** Strict compression before transmission for terrains and selection of Level of Detail - LOD of object or character models represented through the MPEG-4 MeshGrid node.
- ***Texture:*** Decrease of resolution and amount of colours, texture format modifications and compression level.
- ***Video:*** Decrease of frame rate, resolution, colors, compression modifications, elimination of frames, preservation of audio channel only;
- ***Voice messages, audio, audio in video:*** Decrease of the number of output channels, signal sample rate, number of bits for codification and modification in compression and format.

5. Evaluation of the *3DVEAdapt* Framework

In order to evaluate the *3DVEAdapt* framework, a 3D game was used as example. A First person shooter game - FPS entitled "Moving Target" was built, in which the player´s goal is to hit a target driven by a program. The game has a reasonable quality in terms of graphics and audio. The goal is to be able to play the game in mobile and/or low resource devices. The game was implemented with the following functions: avatar movimentation, collision detection, pre-determined behaviour of NPCs. The terrain is a version of the geometric representation of the Quake III Arena [26] converted to MPEG-4. The game was built using the Quake III Arena editor. Their textures were also used keeping the original format. The other components were programmed in Java. The game was performed using the current MPEG-4 3D player version available. The plataform used was an AMD Duron 1,2 Ghz, 256 MB of RAM, Graphic board Geforce 2 MX 200 32 MB, AGP 4X.

It is given below the paths followed for the evaluation of the framework:

1. A download is made of the MPEG-4 file (.mp4) from the Server (3D VE content provider). This file contains the address (URL) of the MSC responsible for controlling the game session;
2. The MPEG-4 Player, when processing the file, initiates the download of the MPEG-J application as well as the framework components (in case the client terminal does not have them already loaded). Both are received as MPEG-lets – originally, these MPEG-lets are in the same machine of the CA and MSC (but not necessarily);
3. MPEG-J application starts the PS and RM framework components;
4. PS examines the device capabilities and send the result to the MPEG-J application;
5. When receiving the data from PS, the MPEG-J application decides on whether there is a need for an initial adaptation of the media that will be received from the CA and what adaptation methods best fits the need. If so, a requirement is sent to the CA;
6. The MPEG-J application starts the download of the game. This request is made to the CA, which delivers to the client terminal an adapted version, in case that was necessary;
7. The PS peforms its second set of tests, this time with the data from the received application. This data is submitted to adaptation tests in order to check the performance gains each mechanism causes;
8. The user makes his/her preference selection;
9. Based on the user´s preferences and on PS´s second set of tests, the adaptation methods are mapped on both the client terminal and the CA;
10. The game is started;
11. By detecting a decrease in the level of any

resource or the frame rate per second, RM sends this data to the MPEG-J application;

12. When receiving information on resources out of pre-established thresholds, MPEG-J application activates one of the adaptation policies defined in step 9;

13. The user ends the application or enters into another MPEG-4 session. In this case, steps from 6 on are repeated (apart from steps 7 and 8 that are not carried out);

When a user changes from one session to another, a different session controller (MSC) takes over what means a change in the CA as well. When a user sends an update message, there is no adaptation to be made – the message is broadcast to the client terminals of other participating users, through the scene synchronization component, named MBK *(Mutech Bookeeper),* briefly described in section 3. When the user sends a voice or video message, the media is captured by the MBK, which forwards the message to the CA, along with the client terminal IDs that should receive that media. The adaptations are made in parallel, and when finished, they are delivered to the destination client terminals.

In step 9, the PS tests all adaptation methods available at the client terminal, i.e., it tests the configuration of the available decoders in order to determine the exact influence an adaptation mechanism has on the user platform (client terminal). For that to be possible, the programmer needs to provide a file containing details on how the tests will be realized. In general terms, the test will simulate the user´s actions playing in an application scene. This simulation will be repeated once for each of the available adaptation methods. For each run, it will be extracted the fps rate gain achieved as the result of the activation of that respective adaptation method. The same will be done for the gain in video memory and CPU. Although a major part of the graphics processing is left to the GPU (Graphical Processor Unit), all other resources can have some influence on the decrease of a scene rendering latency.

Texture Adaptation

A test was performed by the PS in order to measure what impact texture adaptation may cause on the client terminal resources. The available decoders in the client terminal allows the limitation of image resolution output to 64x64 pixels, i.e., every image with length or width above 64 pixels will have its dimension reduced to this value before being loaded to memory. The goal is to decrease the necessary image storage space. Two decoder configurations were used: 1) original decoder - decodes the textures in their standard formats (resolutions vary from 32x32 pixels to 512x512 pixels, but most of the textures are in 256x256 pixels); 2) modified decoder - decodes texture of 64x64 pixels – aspect ratio was maintained (for instance, 256x512 pixels image were converted to 32x64 pixels image). In both tests, sound processing was switched off. Figure 2 shows how texture adaptation impacts on the CPU (through the number of frames per second rendered).

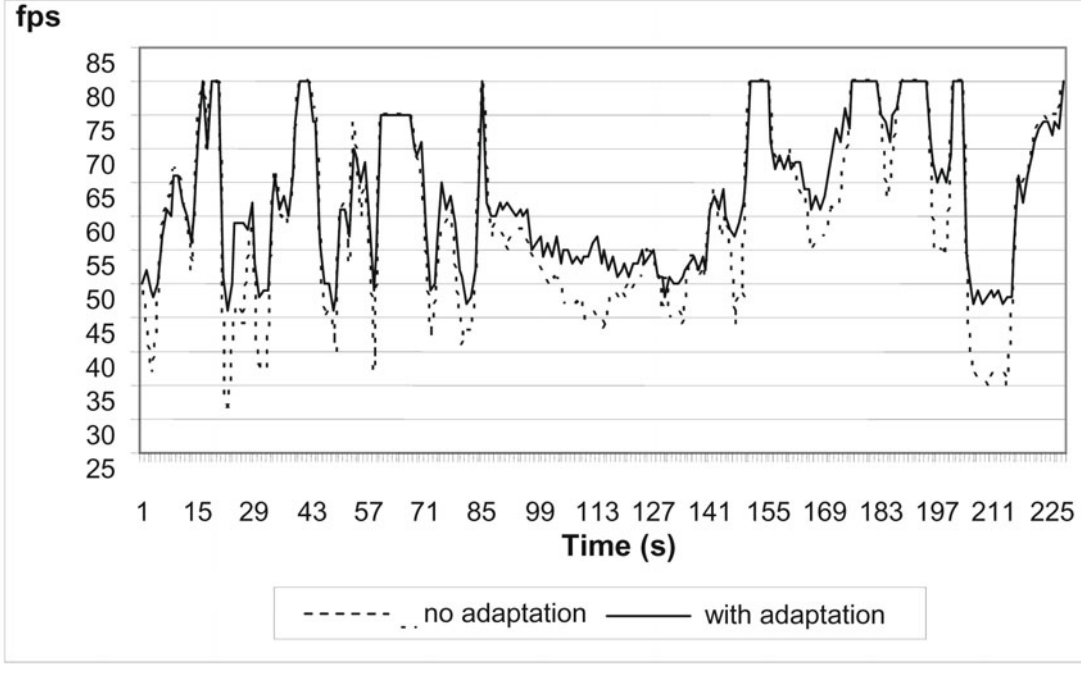

Figure 2: Texture adaptation results.

The largest gains in performance are achieved when a large variety of textures need to be rendered at the same time. The impact on the CPU is influenced by the use and capacity of a GPU (graphics processing unit). One parameter that influences CPU performance is the amount of video and main memories available. The loading time of textures from the main memory to the video memory impacted negatively on the number of fps rendered. In this sense, the tests showed that the use of adaptation caused a reduction of nearly three times the amount of memory necessary to hold the textures. For a low capacity video and main memory device, the adaptation could be realized before the download of the content.

Audio adaptation

In terms of audio, the goal was to reduce the load over the CPU by decreasing the amount of sounds played simultaneously. All sounds used in the tests are PCM format. The files vary from 1 to 55 KB with playing time between 1 and 5 seconds. For the tests, a certain amount of sounds of a certain audio quality are played constantly along the path that is ran through the 3D virtual environment simulation with the user´s actions. Because the files are small, they did not impact significantly on the performance. The first test evaluated the impact of sound quality on the application. Table 1 shows the parameters used.

Table 1: Properties of the sound files used .

Type	Channels	Sample rate (KHz)	Codification Bits	Necessary Network bandwidth (Kbps)
1	1	8	8	88
2	1	16	16	256
3	1	22	16	352

Four series of tests were ran, one per type of audio file used and one for no sound at all (for comparison reasons). The results with 64 sounds were played simultaneously are shown in figure 3.

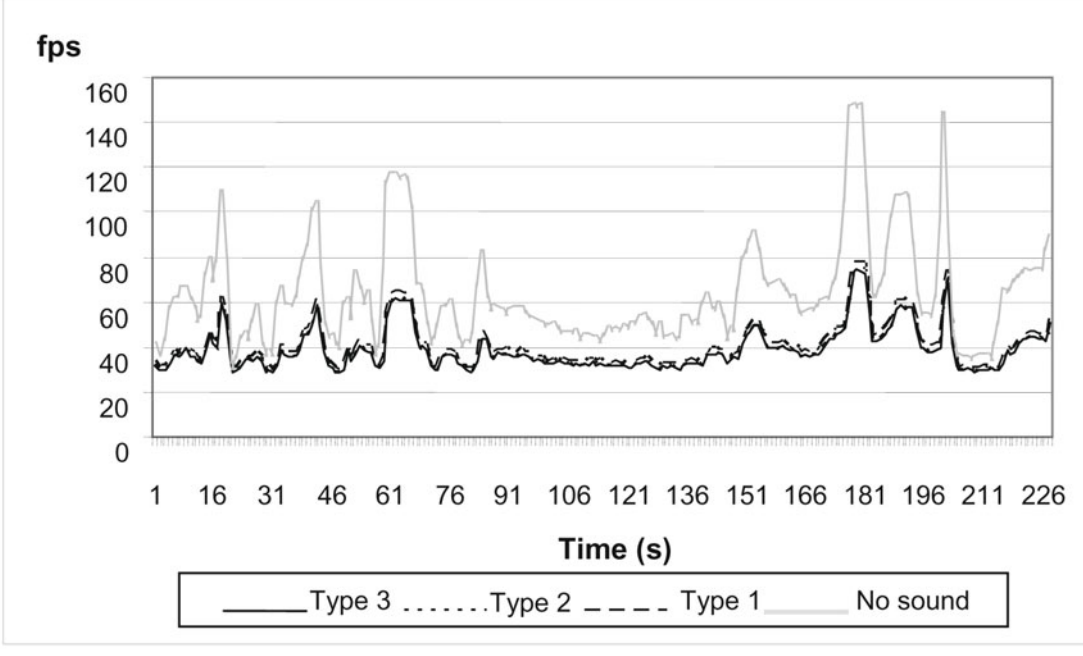

Figure 3: Impact of different types of sound on performance.

It can be observed that the gains are not significative perhaps because the file sizes are small, even for the best quality audio and so, they do not demand large memory. The execution of an adaptation policy to decrease sound quality before downloading the audio files, only makes sense if the device does not support the sound reproduction for a certain sample rate or number of channels.A second test was realized to test the impact a varying number of sounds can cause on a 3D virtual environment. Five series were ran in the simulation environment, each one with 64, 32, 16, 8 and 0 sounds, respectively. All files have quality type 3 (as shown in table 1) and textures have standard quality format. Figure 4 shows the results.

It can be observed gains in all series of tests compared to the serie with 64 sounds. Although the gains do not seem to be significant in this platform, it can be more significative in devices with lower processing capacity. The peak represent less complex areas of the game that are rendered in a much higher rate compared to others. There is no memory gain since all sounds are loaded and what sound is used, and what is not, is decided in real-time, based on the established priority. Table 2 shows the performance average gain measured in fps, compared to 64 sounds processing.

Table 2: Performance average gain compared to tests with 64 sounds.

	32 sounds	16 sounds	8 sounds	No sound
Average (fps)	7	10	12	23
Standard deviation	1.47	2.94	3.1	3.4

The small Standard deviation is due to the concurrency among running processes, which during the test can influence either positively or negatively. An adaptation policy can be triggered by the PS that prevents a low capacity device to play more sounds than it can support. Tests showed that low quality sound results in restrained sound and tone differences are not perceptible. It must be observed that decreasing the amount of sounds played in the 3D environment, can lead to a loss of immersion – the user can see an object but can not hear from it.

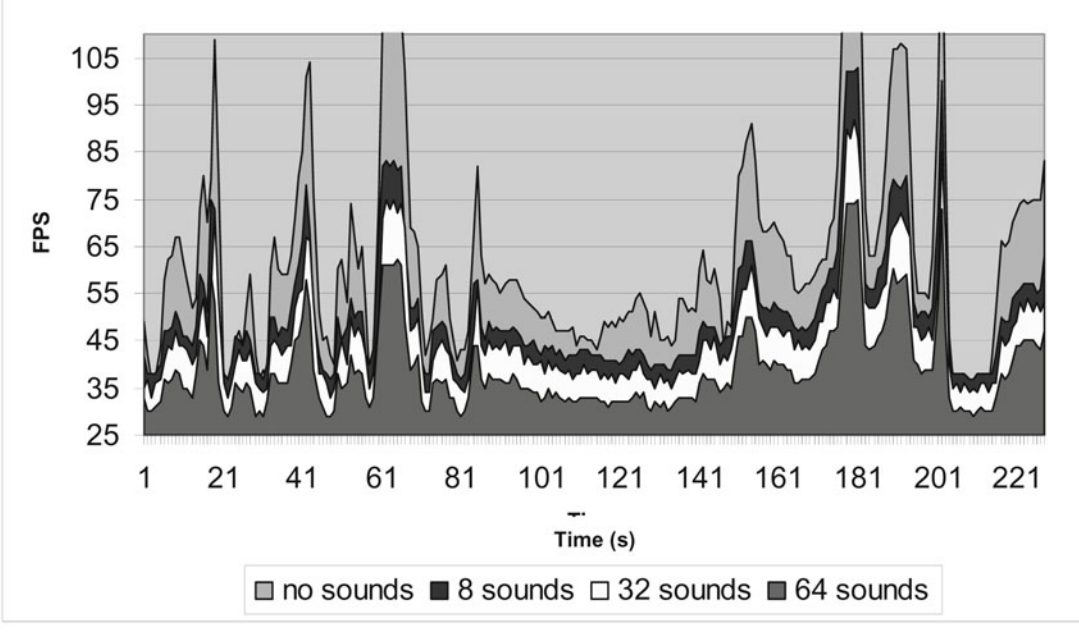

Figure 4: Impact of number of sounds on performance.

6. Conclusions

This work describes the specification and evaluation of the *3DVEAdapt,* a framework that adapts 3D VEs application data and control as a result of monitoring system and application resource level fluctuations. The framework makes decisions whose goals are to keep the performance requested by the user. The framework can be used in MPEG-J applications to help programmers to insert adaptation mechanisms that react to variations in a device level of resources as well as to performance variations of the application itself. The goal of the adaptation is to maintain the performance required by the 3DVE application user in heterogeneous devices. With *3DVEAdapt* framework, the programmer defines the resources to be monitored, as well as the mechanisms that will be activated in order to guarantee the performance level determined by the user. It helps programmers to insert adaptation mechanisms that deal with device and network capabilities heterogeneity; variations in both the user´s environment level of resources and in the scene complexity during 3D VE runtime. The adaptation mechanisms are identified dynamically and used on a per client terminal basis. An advantage of this struture is that it is able to adapt applications to heterogeneous devices, through an evaluation component that identifies, beforehand, if the device can or not process a given 3D VE application. An application that uses the *3DVEAdapt* can free the user from the concerns regarding application behaviour fluctuations by always providing the same performance under different conditions. The framework is integrated to the adaptation specification defined by the MPEG-4 standard. The advantages of integrating the framework to the MPEG-J and multiuser MPEG-4 extensions are twofold: 1) it favors the creation of complex applications, with high degree of interaction, such as multiuser 3D games, training, etc., since applications can be easily integrated to the MPEG-J; 2) because MPEG-4 is aimed at small mobile devices and network narrowband, it is easier to make this solution available to devices from cellular phones to PDAs and set-top-boxes.

A unique aspect of the work is the focus on 3D VE adaptation, as opposed to most of the work found in the literature that deals mostly with video, audio and text (web pages) adaptation. Another differential of this work is the integration to a multiuser structure based on the MPEG-4 standard, where multiple users can exchange not only synchronization messages, through the MSC and MBK components, but also adaptable content among users. The *3DVEAdapt* framework adaptation mechanisms can identify the format used and the existing configurations in real-time, allowing each CA to provide the configurations necessary. The framework was evaluated and presented encouraging results. However, the MPEG-4 player showed low rendering performance, mainly due to the lack of optimization of the freely available version of the MPEG-4 Player (where functionality is more important than performance). The adaptation was shown to be a feasible solution for multiuser 3DVE applications. Further tests have to be made for heterogeneous devices.

References

[1] Boukerche, A. Dzermajko, C. and Araujo, R.B. A Grid-Filtered Region-Based Approach to Support Synchronization in Large Distributed VEs, in preparation, 2004.

[2] Boukerche, A., Araujo, R.B. and Duarte, D.D., *3D Virtual Environments Extensibility through Non-linearInteractive Stories*, accepted to DS-RT, 2004.

[3] Cadmium Project in www-sor.inria.fr/projects/cadmium/index.html, April, 2000.

[4] Capps, M.V. *Fidelity Optimization in Distributed Virtual Environments*. Ph.D. thesis, Naval Postgraduate School, Monterey, CA, 2000.

[5] Chang, F. and Karamcheti, V. *Automatic Configuration and Run-time Adaptation of Distributed Applications.* Nineth IEEE Symposium on High Performance Distributed Computing (HPDC), August 2000.

[6] Coda and Odissey Mobile Information Access, em www.cs.cmu.edu/afs/cs/project/ coda/Web/coda.html, April, 2000.

[7] Eisenstein, J., Vanderdonckt, J. and Puerta, A. *Adapting to Mobile Contexts with User-Interface Modelling.* IEEE Computer, 2000.

[8] Greenhalgh, C., S. Benford and G. Reynard. *A QoS Architecture for Collaborative Virtual Environments*. In Proceedings of ACM Multimedia 1999, pp. 121 – 130.

[9] Helma, J. SIGGRAPH '96 Course #33 James Helman - Silicon Graphics - *Designing Real-Time 3D Graphics for Entertainment.*

[10] http://3dgraphics.about.com/od/web3dtechnologies/

[11] http://www.adobe.com/products/atmosphere/main.html/

[12] http://www.demicron.com

[13] http://www.macromedia.com

[14] http://www.marrin.com/vrml/private/EmmaWhitePaper.html

[15] http://www.sait.samsung.co.kr/snhc

[16] http://www.vrml.org/WorkingGroups/living-worlds/draft_2/index.htm.

[17] http://www.web3d.org

[18] http://www.xj3d.org/

[19] http://www/vrml.org
[20] ISO/IEC 14496-1:2001#38 Information technology — Coding of audio-visual objects, Part 1:Systems, August 2001.
[21] Lara, E., Wallach D. S. and Zwaenepoel, W. *Puppeteer: Component-based Adaptation for Mobile Computing*, Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California, in http://www.cs.rice.edu/~delara/papers.
[22] Martim, Ioana M. ARTE - *An Adaptive Rendering and Transmission Environment for 3D Graphics* - IBM T. J. Watson Research Center, 2001.
[23] N4264 - *Overview of MPEG-4 Standard* (V.18 – Singapore Version), 2001.
[24] Overmars, M. *It's All in the Game* - Game Developer, August 2000, pp. 24-32.
[25] Pulse 3D - www.pulse3d.com/pulse/
[26] Quake III Arena – Developed by Id Software, Distributed by Activision. PC GAME - US RELEASE, 1999
[27] QuickTime VR - www.apple.com/quicktime/qtvr/
[28] Rao, D. *Efficient and Portable Middleware for application-level Adaptation*, Master of Science thesis, Department of Computer Science and Applications Virginia Tech, Blacksburg, Virginia. 2001.
[29] The Java 3D API Specification. - https://java3d.dev.java.net/.
[30] Trefftz, H., Marsic, I. and Zyda, M. *Handling Heterogeneity in Networked Virtual Environments,* IEEE Virtual Reality 2002 Conference. Published in Presence – Teleoperators and Virtual Environments, Vol. 12, Issue 1, 2002.
[31] Unreal 2002 – Unreal technology in http://unreal.epicgames.com/
[32] W4415- ISO/IEC JTCL/SC29/WG11 – N4415 – MPEG-4 Systems. MPEG, ISO. Pattaya, December, 2001.