Adaptive 3D Content for Multi-Platform On-Line Games

Francisco Morán[1], Marius Preda[2], Gauthier Lafruit[3], Paulo Villegas[4] and Robert-Paul Berretty[5]
[1] *Universidad Politécnica de Madrid, ES* [2] *Institut National des Télécommunications, FR*
[3] *Interuniversitair Micro Electronica Centrum, BE* [4] *Telefónica I+D, ES* [5] *Philips, NL*
[1] *Francisco.Moran@upm.es* [2] *Marius.Preda@int-evry.fr* [3] *Gauthier.Lafruit@imec.be*
[4] *Paulo@tid.es* [5] *Robert-Paul.Berretty@philips.com*

Abstract

Most current multi-player 3D games can only be played on dedicated platforms, requiring specifically designed content and communication over a predefined network. To overcome these limitations, the OLGA (On-Line GAming) consortium has devised a framework to develop distributive, multi-player 3D games. Scalability at the level of content, platforms and networks is exploited to achieve the best trade-offs between complexity and quality. Besides, standardized content compression formats (MPEG-4, JPEG 2000) are used in OLGA's framework, enabling easy deployment over existing infrastructure, while keeping hooks to well-established practices in the game industry.

1. Introduction

OLGA (www.ist-olga.org) is the short name of a research project partially funded, from April 2004 to September 2006, by the European Commission under FP6-IST (cordis.europa.eu/ist). Its full name is "A unified scalable framework for **O**n-**L**ine **GA**ming", as its ultimate goal is to provide a framework for developing **scalable** 4D (animated 3D) game content that could be adaptively streamed to a variety of terminals over heterogeneous networks; and to do so by using (and improving, whenever possible) de jure international **standard** coding formats such as MPEG-4 AFX (Animated Framework eXtension) [8].

Thanks to OLGA's scalable 4D content authoring and compression tools, it is possible to render the same textured 4D content at wildly different qualities and frame rates, according to each network and terminal profile. Figure 1 gives an idea of how OLGA's game test bed, named GOAL, runs on a PC (Personal Computer) and a CP (Cell Phone). We decided as well to enable the players to publish their own 4D content for its use in the game: OLGA's tools are not only provided to game designers, but also to end users.

Figure 1. Screen shots from both the PC and CP versions of GOAL, OLGA's game.

Section 2 elaborates on OLGA's tools and explains how scalable coding can be exploited for adapting the execution time in a specific terminal, under excellent quality vs. bit-rate vs. memory vs. execution time trade-offs of the 3D geometry, textures and animation. But OLGA's mission was not only producing scalable 4D content authoring and compression tools. Another two of its main goals were to deploy a scalable game platform (an infrastructure consisting of both servers and network) that would adapt content in a distributed way, and to provide a set of terminals to validate OLGA's technology by implementing GOAL on them. Sections 3 and 4 comment on how those other OLGA targets were achieved. Finally, Section 5 concludes our presentation.

2. Standard scalable 4D content

A few years ago, high quality 3D graphics were a crucial asset for making a computer game successful. Nowadays, they are practically taken for granted: for current players, 4D content looking great is not a bonus but nearly a must. And creating compelling 4D objects and characters is a very time-consuming task even when that content is not scalable.

A key ingredient of OLGA is its software toolset for content creation, conversion and compression, which provides game designers, as well as end users, with flexible solutions to create scalable 4D content from scratch, or to recycle already existing 4D content to have it be scalable, and to compress it efficiently. Scalable (off-line) coding is of the utmost importance for OLGA to enable the continuous adaptation (at run-time, under constrained system resources) of the 4D content parameters, so that the best trade-off between instantaneous 3D rendering quality and animation speed can be achieved. Such adaptation is possible thanks to progressive bit-streams that can be stripped through packet selection mechanisms for view-dependent decoding (or even streaming) scenarios, in which only the visible portions of a 3D object geometry and texture are transmitted and decoded at the appropriate quality. The animation quality can also be scaled by performing only those transformations yielding a visible effect for the player.

Another key ingredient of OLGA, besides scalability, was compliance to the maximum possible extent to international standards. MPEG-4 was chosen because it already featured the following tools for scalable 4D content when OLGA started:

- **3D geometry** (see Section 2.1): among the several tools targeting the compression of polygonal meshes in MPEG-4 AFX, we chose the one based on WSSs (Wavelet Subdivision Surfaces).
- **2D textures** (see Section 2.2): both MPEG-4's native format for textures, VTC (Visual Texture Coding), and JPEG 2000 are also wavelet-based. We chose JPEG 2000, but made sure MPEG-4 would support it.
- **Animation** (see Section 2.3): BBA (Bone-Based Animation) is a sub-toolset of MPEG-4 AFX permitting to animate generic articulated characters based on the "skeleton and skin" paradigm.

2.1. 3D geometry

Several 3ds Max plug-ins were implemented to enable an artist to automatically simplify an arbitrary connectivity 3D mesh, remesh it to have subdivision connectivity (see Figure 2), and code it in a scalable manner:

- Our 3D mesh simplification plug-in for 3ds Max, olgaQAttSimp, is based on the QEM (Quadric Error Metrics) technique [4] and yields significant improvements over 3ds Max's native Optimize: the geometry obtained is much more efficient (in terms of triangle count for a given approximation error) and the texture coordinates are correctly handled. Compared to the MultiRes modifier that comes also standard with the newer versions of 3ds Max, olgaQAttSimp is very efficient when it comes to smooth content, and roughly equivalent for "not-well-rounded" shapes. But, in all cases, it allows the artist to control more closely the mesh decimation and obtain more subjectively faithful final results by selecting certain regions to be preserved. A simplificator software module has also been developed based on olgaQAttSimp, and integrated in the LCSs (Local Content Servers: see Section 3) to allow run-time vertex removal.

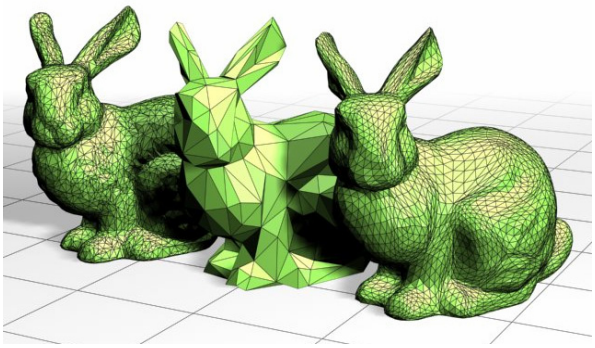

Figure 2. An arbitrary connectivity mesh (left) modeled as a WSS after base mesh extraction (middle) and remeshing with subdivision connectivity (right).

- The coding can comply to the WSS tool already in MPEG-4 AFX, a.k.a. "WaveSurf" [9], or follow the PLTW (Progressive Lower Trees of Wavelet coefficients) technique [1], explained below and proposed to MPEG for its adoption in a future Amendment of AFX. The two corresponding decoders (PLTW-based and MPEG-4-compliant) are both integrated in OLGA's software framework for the PC platform. As for the CP platform, only the PLTW-based decoder has been ported to Symbian OS, since it has less memory requirements than WaveSurf.

2.1.1. Geometry quality/bit-rate/memory trade-off

Once a 3D shape is modeled as a WSS, it is fit for multi-resolution coding. Our research in this field focused on new methods that could be more suitable for resource-limited devices than the SPIHT-based ones, like the WaveSurf tool. For a decade already, the SPIHT (Set Partitioning In Hierarchical Trees) technique has been the reference against which to compare other coding techniques based on the wavelet transform. The problem of SPIHT is that, although its bit-streams are SNR scalable, they are not spatially scalable, and cannot be easily parsed according to a given maximum resolution (i.e., number of pixels or triangles) or LOD (Level Of Detail) tolerated by the decoder. There is little point in encoding a 3D mesh with

thousands of triangles if the CP that must render it can barely handle hundreds. Furthermore, from the memory viewpoint, having a perfectly SNR scalable bit-stream that may have bits corresponding to details of LOD 3 before those of LOD 1 makes also little sense, as the decoding process alone will completely eat up all the CP resources: even if memory is not allocated for the triangles of LOD 3 (which will never be rendered), their detail trees must be created to follow the SPIHT algorithm.

The main novelty of the PLTW technique [1] is that the resulting bit-stream does not impose on the less powerful decoders the need of building detail trees as deep as required by the maximum LOD encoded, because the wavelet coefficients are sent on a per-LOD basis, thus achieving "local SNR scalability" within "global spatial scalability". With PLTW, the set of coefficients is also hierarchically traversed, but they are scanned in LODs, which yields a spatially scalable bit-stream. The decoder first receives all the coefficients corresponding to a LOD and, only when it has finished reading them, it proceeds (if it has enough resources) with those from the next. However, thanks to bit-plane encoding, bits from each LOD are ordered in such a way that the first to arrive are the ones that contribute more to lower the reconstruction error, while bits from negligible coefficients arrive last.

A comparison of our PLTW coder vs. two other SPIHT-based coders is illustrated by Figure 3, which plots, for two different 3D models, the rate distortion curves for: *i*) our PLTW coder, which does include AC (Arithmetic Coding) as a final step; *ii*) a version of the SPIHT algorithm with AC; and *iii*) the WaveSurf tool of MPEG-4, which also uses SPIHT, but without AC. Except at very low rates, where PLTW is still reconstructing upper LODs and does not benefit from the smoothing effect of subdivision (while its competitors do), PLTW always results in higher PSNRs for the same bit-rate. It is also noticeable how none of the SPIHT-based coders is able to reach the same PSNR as the PLTW coder even employing 160% (SPIHT-AC) or 330% (MPEG-4) of the bits used by PLTW for the same quantization set of values. The poor results of the WaveSurf coder are mostly due to the overhead introduced to support view-dependent transmission of coefficient trees.

2.1.2. Geometry quality/bit-rate/run time trade-off

The use of compressed, multi-resolution content enables the adaptation of its complexity (and hence also its visual quality) to the available bandwidth and terminal resources. WSSs permit to code the shape of a 3D model in a multi-resolution manner with very good compression, but require a large CPU overhead for a fine-grained, on-the-fly control of the content complexity in execution time regulated applications such as networked, interactive 3D games. In fact, the CPU overhead for controlling the execution time with MPEG-4's WaveSurf tool is sometimes as large as the 3D graphics rendering execution time itself.

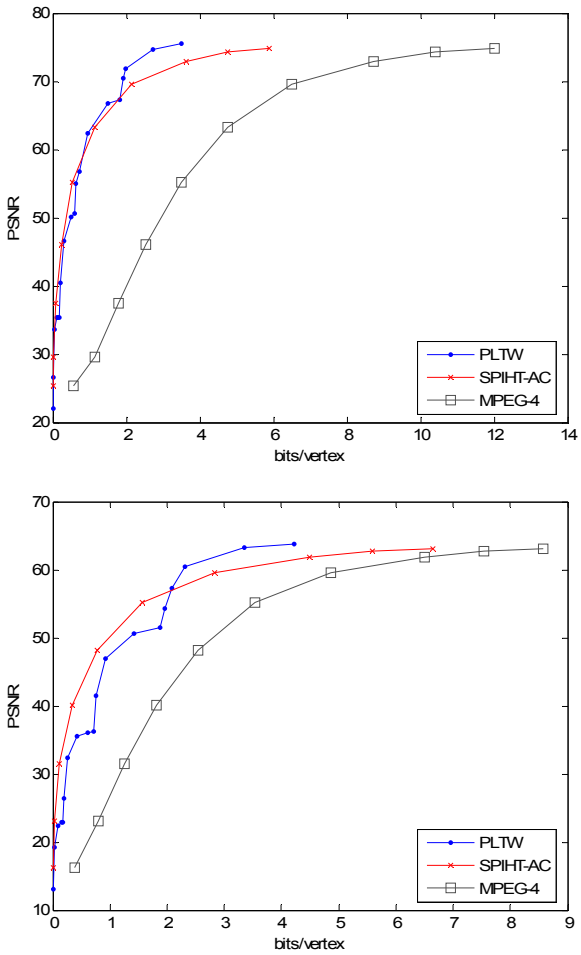

Figure 3. PLTW vs. SPIHT and MPEG-4's WaveSurf for the Max Planck (top) and bunny (bottom) models.

Moreover, typical implementations of WSSs multiply by four the number of triangles in every subdivision step, which enables only very discrete LOD management, and therefore yields abrupt and often disturbing quality changes while only supporting coarse-grained adaptation to a target execution time. Besides improving the compression efficiency and the adequacy to weak terminals with the PLTW technique, we introduced some add-ons to enable a low-complexity, yet efficient fine-grained quality/run time trade-off in execution time control.

To achieve this target, the WSS mesh regions are progressively decoded in a continuous LOD fashion, by subdividing only the important regions of the ge-

ometry. The importance and order for subdividing the triangles is given by their impact on the error to the target mesh, i.e. the triangles that decrease this error the most are subdivided first. These non-uniformly subdivided meshes allow a fine-grained control of the resolution of the geometry, resulting in small variations of the visual quality while achieving a target execution time. With special subdivision platform mapping techniques using LOD-based moving windows [13], the complexity of the subdivision control is largely reduced, resulting in an overhead of only a small percentage in the final decoding and rendering execution time for two different platforms: a high-end PC and a low-end CP.

In order to actually steer the execution time control, the execution time, and especially the rendering time, should be estimated for a large range of triangle budgets. We have used previously reported performance models for the software and hardware rendering pipelines [12], according to which the most important parameters are the number V of processed vertices (for the vertex processing) and the number F of fragments (for the rasterizing); additional parameters important for the software model are the number S of spans and the number T of visible triangles. The coefficients of the performance model are derived with an off-line calibration procedure that first measures on the device the rendering time for many different objects with different sizes (F) and complexity (V and T), and then computes the average values of the coefficients c_α ($\alpha \in \{T, F, S\}$) with multi-linear regression analysis.

2.2. 2D textures

After carrying out a preliminary comparative study between JPEG, JPEG 2000 [5] and MPEG-4's VTC [7] with respect to the considered criteria and desired functionalities within OLGA, the JPEG 2000 technology was selected, and several tools developed:

- A plug-in enables 3ds Max to import and export JPEG 2000-compliant textures (at the time of writing, the last version of 3ds Max, nr. 9, did not support natively JPEG 2000 yet).
- Tools enabling view-dependent texture streaming thanks to JPEG 2000 and JPIP (JPEG 2000 Internet Protocol), in which a bit-stream packet selection mechanism takes the user's viewpoint information into account. Implementations were made for both PC and CP, and both the JPEG 2000 and JPIP decoders were optimized towards their usage in a 3D graphics texture context, and extended with additional control tools tailored to a view-dependent texture streaming scenario. The JPIP cache mechanism is adapted to minimize the CP memory usage.
- A JPEG 2000 bit-stream packet selector has been integrated in the simplificator module running on LCSs (see Section 3), that supports resolution scaling and bit-plane removal. The LOD selection takes into account both the available bandwidth between LCS and terminal, and the terminal screen resolution.

But OLGA's most important contribution with respect to textures has little to do with JPEG 2000 (except for having succeeded at having MPEG-4 support it as one of its native image formats), as in fact the work described above mostly consisted in implementing and porting already existing algorithms and software. At least conceptually, OLGA's main contribution was detecting drawbacks in the current IFS (Indexed Face Set) tool of MPEG-4, inherited from VRML97, and defining the so-called "IFS++" format for 4D meshes with enriched vertex attributes such as multiple texture coordinates and bone-vertex influence coefficients. This activity led to another MPEG proposal, which will hopefully be included as well in a future AFX Amendment.

2.2.1. Geometry+texture quality/bit-rate trade-off

Besides the execution time variation with the platform and content parameters [12], the linearity of the cost with the object parameters was also observed in the bit-rate of the textured MPEG-4 objects: with a regression coefficient of 93% measured over 60 objects, the original MPEG-4 file size s decreases roughly bilinearly with the JPEG 2000 texture LOD (with negative slope m_1) and the object mesh LOD (with negative slope m_2). Small file sizes s with large (absolute values of) m_1 and m_2 correspond to small bit-rates that decrease very rapidly with decreasing LOD: the corresponding objects representing only a small fraction of the total bit-rate at all LOD levels, they have low priority to be scaled for global (over all objects) bit-rate adaptation. On the other extreme, large s with small m_1 and m_2 correspond to large bit-rates that decrease very slowly with decreasing LOD, hence representing barely any opportunity of down-scaling for global bit-rate adaptation. Consequently, large s with large m_1 and/or m_2 are the most appealing candidates for bit-rate adaptations: starting from a large full resolution bit-rate contribution, they scale very well by adjusting the texture and/or mesh LOD.

Together with the improvements introduced by the geometry and animation coding tools, a global quality/bit-rate/execution time control can be obtained over all objects. The details of this intelligent global adaptation are beyond the scope of this paper, since it mainly consists in finding heuristics for approximately solving an NP-hard knapsack problem [3].

2.3. Animation

Virtual characters are the most complex objects in a 3D game, and OLGA's main vision, using scalable content within a standardized framework, was also applied to them. We used as a basis the BBA (Bone-Based Animation) specification [10], a subset of MPEG-4 AFX defining a framework for representing and animating skinned models. On top of the generic compression used for the object and scene graphs, which is based on MPEG-4 BIFS (BInary Format for Scenes) [6], BBA defines a compressed representation of the animation parameters: bone transforms, muscle deformations and morphing weights.

2.3.1. Animation quality/bit-rate trade-off

To represent compactly the data required by the animation of textured 3D models (varying vertex attributes: essentially spatial coordinates but also normals or texture coordinates), some kind of redundancy in the animation is usually exploited: either temporal, and then linear or higher order interpolation is used to obtain the value of the desired attribute between its sampled value at certain key frames; or spatial, and then nearby vertices are clustered and a unique value or transform is assigned to each cluster. MPEG standardized an approach for compression of generic interpolated data [6], able to represent coordinates and normal interpolation. While generic, this approach does not exploit the spatial redundancy. Concerning avatar animation, one of the most used animation content for games, a subset of MPEG-4 named FBA (Face and Body Animation) [7] allows compression at very low bit-rates. However, FBA imposes a rigid definition of the avatar and the difficulty to set up the proposed deformation model. At the time the OLGA project started, we were in the final stage of standardizing BBA, an extension of FBA within MPEG-4 AFX.

BBA allows to represent animated, generic 3D objects based on the skin and bones paradigm, and to transmit the animation data at very low bit-rates by exploiting both the temporal and spatial redundancies of the animation signal. Within OLGA, we addressed the terminal/network adaptation, compression and rendering of BBA-based content. We considered the adaptation of animated content at two levels: geometry simplification constrained by dynamic behavior [11] and animation frame reduction. The dynamic behavior was expressed as constraints used to parameterize the QEM technique [4]. We introduced a weighting factor to specify how a given set of bones influences the simplification procedure. The biomechanical characteristics (i.e., the relationships between skin and bones) were directly exploited to constrain and control the simplification procedure. We applied the developed algorithm to OLGA animated objects, previously converted into MPEG-4-compliant skinned models. Figure 4 shows the comparative results of animated model simplification for the developed approach, called AC-QEM, vs. plain QEM.

QEM-simplified model (491 vertices) AC-QEM-simplified model (497 vertices)

Figure 4. AC-QEM vs. QEM: qualitative results for the dragon model.

Decoding and rendering animation data on small memory devices such as CPs requires server-side animation adaptation. Our approach was to reduce the number of the animation key frames so that the CP must only store a small quantity of information and use temporal interpolation. Animation simplification based on frame reduction was achieved by considering a progressive approach. Given an original animation sequence of n frames, to obtain a simplified sequence with $m < n$ frames approximating the original curve, the area between the original curve and the reconstructed one must be minimized. Considering this condition for all bones (or the subset of extreme bones), the optimization problem becomes difficult to solve. To overcome the complexity, we adopted an incremental approach: for each pair of three frames and for each extreme bone, we compute the area between the original signal and the one reconstructed by linear interpolation. We sum these areas for all extreme bones and the minimum of the sums indicates the frame that has to be removed. We repeat the algorithm until the number of removed frames equals $n - m$. After frame reduction, a new BBA stream is obtained by encoding the m frames, and indicating for each frame the number of intermediate frames to be obtained by interpolation on the terminal.

2.4. Complete 4D scene exporting in MPEG-4

Finally, three 3ds Max plug-ins have been released that are able to export whole scenes containing several 4D objects: the first exports fully MPEG-4-

compliant textual (*.xmt) and binary (*.mp4) files; the second exports MPEG-4-compliant textual (*.txt) for animated characters: object graph definition and animation data; and the third outputs bit-streams that are not yet fully MPEG-4-compliant in that they follow OLGA's IFS++ format and the PLTW-based coding of WSSs if the user so wishes.

3. Servers and network

The work related to servers and networks comprised the design, development and testing activities for the integration of the game test bed versions with the various versions of the network architecture. Both the PC and CP clients communicate and authenticate with a central lobby server, which manages a distributed network of game logic servers, called ZGSs (Zone Game Servers), and content adaptation and delivery servers, called LCSs (Local Content Servers), as opposed to the GCS (Global Content Server), which is a centralized resource, like the lobby server.

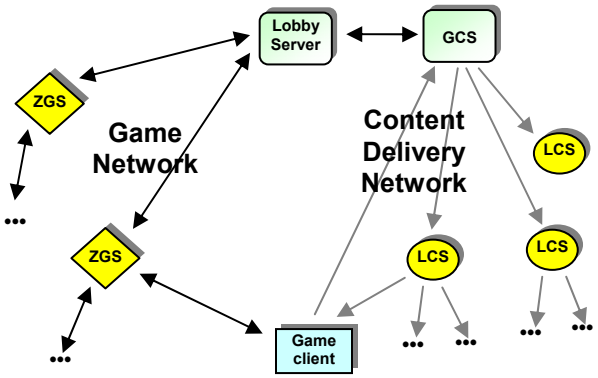

Figure 5. Network architecture.

Figure 5 illustrates OLGA's network architecture, and the conceptual decoupling of the game network and the content delivery network. Load balancing and recovery mechanisms for these distributed networks of servers were implemented and successfully tested. The ZGSs have basic gaming functionality and can handle both player avatars and non-player characters, and both static and dynamic content. Instead of using a completely centralized solution, or one with a grid of homogeneous servers, we decided to have many heterogeneous ZGSs and LCSs, potentially hosted at the most powerful PCs of the players themselves. This allows a high degree of network scalability against the number of clients.

3.1. Content delivery network

Sending or updating game content (i.e., objects to be rendered) over the network is not a frequently used option, although multi-player on-line games pushing content through the network instead of locally storing all data do exist. However, most of these games reduce a priori the transmission bandwidth by subdividing the world in sub-worlds (3D tiles) and referencing pre-stored items, and texture data is seldom transmitted. We chose instead to enable live update, distribution and adaptation of content.

This adaptation requires extensive CPU power and memory. It is not practical to serve dynamically rendered content using a pure client-server architecture, and that is why the P2P (Peer-to-Peer) model was chosen. The final key features of OLGA's content delivery network are:

- The system works through very heterogeneous networks and terminals, from high-end 3D graphic PCs connected to broadband Internet to mobile handsets connecting through 3G networks, all simultaneously active in the same game, and interacting with each other.
- The LCSs are not passive distribution nodes: on the contrary, they actively adapt the content to the client characteristics before delivery. Adaptation is done through the set of simplification tools described in Section 2, and the LCSs cache the result of simplifications to save processing effort.
- A content adaptation server is installed on every client PC, and may be called upon dynamically by the Lobby Server to act as an LCS, depending on system conditions.
- The amount of available content in the game is variable, and can be updated from all nodes in the network. Any game client can create its own content (in standard formats) and insert it into the game in real time, by uploading it to the GCS and using the ZGS to add a reference to the new content in the game; other players will download the content from their LCS as needed by game information provided by the ZGS.

Given the P2P properties of the content delivery network, some scalability is inherent to it: new clients entering the game also bring new servers, thus leveling the capacity of the network. However, as playing the game makes the clients behave unpredictably from the point of view of content requests, a means of ensuring dynamic adaptation to changing conditions is necessary and was implemented, although it is out of the scope of the present paper.

4. Multi-platform 3D rendering

As for the terminals, OLGA supports a variety of them, which were used within the project to test and

validate the scalability of game content. The main focus was on the realm of PCs, ranging from high-end gaming ones to laptops, but also mobile terminals were used. Two software platforms were produced:

- GOAL, our game test bed, is available both on MS-Windows-based PCs and on CPs running Symbian OS v8 and supporting J2ME, notably the Nokia 6630. Game logic was implemented on both versions of the game, and decoders integrated for the simplified content downloaded from the network. For the CP, a part of the software is programmed in Java, and the content decoders are programmed in Symbian, the Symbian framework being connected through a socket with the Java game engine. Rendering of the final graphics is done in software on the ARM embedded in the OMAP processor of the Nokia 6630. For screen shots of GOAL on both types of terminals, see Figure 1.

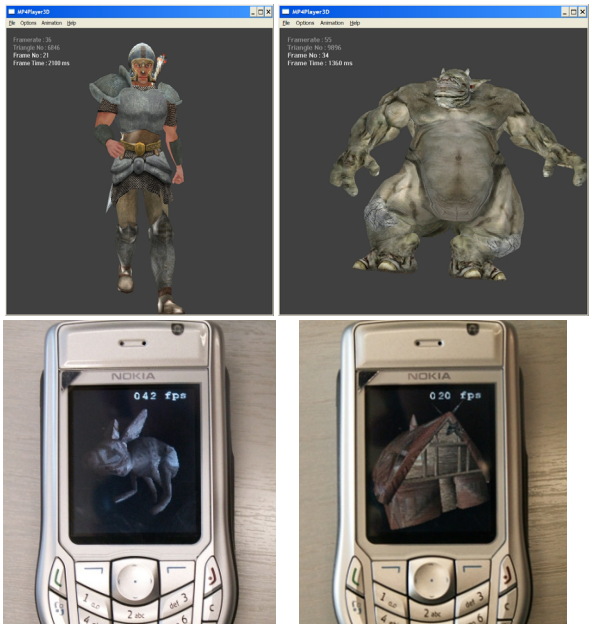

Figure 6. OLGA models loaded in the PC (top) and CP (bottom) versions of the MPEG-4 player.

- Besides, we developed a stand-alone MPEG-4 player to visualize textured 4D content on both PCs and CPs: see Figure 6. We selected a small number of the scene graph nodes defined in the BIFS specification, which is enough to represent static and animated textured 3D objects. Then, starting from an open source BIFS decoder named GPAC (gpac.sourceforge.net), we derived a simplified BIFS decoder by implementing just the selected nodes. To allow texture mapping, we plugged in both JPEG and JPEG 2000 decoders and, to support animation, we optimized the initial BBA decoder we had developed for PC and also ported it to Symbian OS v8. Finally, we developed the rendering layer by using DirectX 9 for PC and OpenGL ES for CP.

4.1. 3D displays

Special attention was given to the final rendering of 3D gaming content. Nowadays, new terminal and display developments drive new applications such as real 3D viewing. It is possible to design lenticular sheets to turn a flat, 2D matrix display into an auto-stereoscopic multi-view display able to present the viewer with different images from various slightly different viewing angles. A format highly suitable for content transmission for such multi-view displays is video enriched with depth information. The format allows for minor displacements of foreground objects with respect to background scenery that are needed to present the viewer with the required different images [2].

Figure 7. GOAL screen shot:
image and depth buffers (left), and actual image on auto-stereoscopic 3D display (right).

To make the game experience more immersive, we endowed some terminals with such auto-stereoscopic 3D displays. Although, in theory, multiple images from different viewpoints can be rendered individually on the device, this solution is not optimal. From both the bandwidth and computational complexity points of view, it is desirable to render only one viewpoint and provide the depths of the pixels in the computed view to the display. Subsequently, a dedicated processor in the display can render the desired viewpoints at high quality [2]. In the GOAL terminals, we adopted the latter approach, and provide the depth information that is available in the *z*-buffer of the GPU to the 3D display. Therefore, the 3D content is transferred through the OLGA framework to the device, subsequently used to render the scene according to the current game status. Next, the frame and depth information are transferred to the 3D display, which renders the final scene in multi-view 3D, as suggested by Figure 7.

5. Conclusions

Today's multi-player 3D games often rely on dedicated/ proprietary technological solutions for their servers (e.g., massively parallel, brute-force grid computing), and scale down content a priori, according to the bandwidth or rendering power of the "weakest" node in the infrastructure. The OLGA (On-Line GAming) consortium opted for a completely different paradigm: exploiting the scalability at the level of content, platforms and networks, possibly adapting the content, network and processing load to the distributive resources available over the end-to-end delivery chain. OLGA's 4D (animated 3D) content is not stored locally on one single server or local storage medium (e.g., DVD), but is rather distributed over a multitude of servers spread all over the network with adequate load-balancing and fault-tolerance policies, and possibly hosted at the most powerful PCs of the players themselves!

The 4D content is actively pushed from the available servers to the gaming terminals but, since OLGA's 4D content authoring and compression tools are provided to end users as well as to game designers, the players can develop and publish their own content, which then becomes part of the persistent world, and benefits from OLGA's standardized framework for adapting scalable content to the varying processing and bandwidth capacities of a heterogeneous infrastructure, and to the very different rendering power of heterogeneous terminals. OLGA's 4D content authoring and compression tools do not impose constraints on the content complexity: game developers and players are free in their creativity, and OLGA's tools take care to adapt to any circumstances — not the other way around, as is usually the case…

We managed to integrate a chain of content conversion, transmission and rendering technologies into a heterogeneous infrastructure and terminal set, demonstrating real-time interactive 4D content adaptation. We developed a distributive multi-player 4D game but, more importantly, we developed a framework to develop distributive multi-player 4D games, or other multimedia applications with heavy and highly variable bandwidth and rendering requirements. And our framework hooks to a complete toolkit of standardized content representation/compression formats (MPEG-4, JPEG 2000), enabling easy deployment over existing infrastructure, while not impeding well-established practices in the game development industry.

6. References

[1] M. Avilés, F. Morán and N. García: "Progressive Lower Trees of Wavelet Coefficients: Efficient Spatial and SNR Scalable Coding of 3D Models", Proc. Pacific-rim Conf. on Multimedia, Springer LNCS vol. 3767, p. 61-72, November 2005.

[2] R.-P. M. Berretty, F. J. Peters and G. T. G. Volleberg: "Real Time Rendering for Multiview Autostereoscopic Displays", Proc. Stereoscopic Displays and Applications Conference, SPIE vol. 6055, p. 208-219, January 2006.

[3] J. Bormans, N. Pham Ngoc, G. Deconinck and G. Lafruit: "Terminal QoS: Advanced Resource Management for Cost Effective Multimedia Applications", chapter (p. 183-201) of "Ambient Intelligence: Impact on Embedded System Design", Kluwer, 2003.

[4] M. Garland and P. S. Heckbert, "Surface Simplification Using Quadric Error Metrics", Proc. ACM SIGGRAPH, p. 209-216, August 1997.

[5] ISO/IEC JTC1/SC29/WG1, a.k.a. JPEG (Joint Photographic Experts Group): "Standard 15444-1", a.k.a. "JPEG 2000 Part 1: Core coding system", 2004.

[6] ISO/IEC JTC1/SC29/WG11, a.k.a. MPEG (Moving Picture Experts Group): "Standard 14496-1", a.k.a. "MPEG-4 Part 1: Systems", 1999.

[7] ISO/IEC JTC1/SC29/WG11: "Standard 14496-2", a.k.a. "MPEG-4 Part 2: Visual", 1999.

[8] ISO/IEC JTC1/SC29/WG11: "Standard 14496-16", a.k.a. "MPEG-4 Part 16: Animation Framework eXtension (AFX)", 2004.

[9] F. Morán and N. García: "Comparison of Wavelet-Based 3D Model Coding Techniques", IEEE Tr. Circuits and Systems for Video Technology, vol. 14-7, p. 937-949, July 2004.

[10] M. Preda and F. Prêteux: "Virtual Character within MPEG-4 AFX", IEEE Tr. Circuits and Systems for Video Technology, vol. 14-7, p. 975-988, July 2004.

[11] M. Preda, S. Tran and F. Prêteux: "Adaptation of Quadric Metric Simplification to MPEG-4 Animated Object", Proc. Pacific-rim Conf. on Multimedia, Springer LNCS vol. 3767, p. 49-60, November 2005.

[12] N. Tack, F. Morán, G. Lafruit and R. Lauwereins: "3D Graphics Rendering Time Modeling and Control for Mobile Terminals", Proc. ACM Web3D Symposium, p. 109-117, April 2004.

[13] K. Tack, G. Lafruit, F. Catthoor and R. Lauwereins: "Eliminating CPU Overhead for On-the-fly Content Adaptation with MPEG-4 Wavelet Subdivision Surfaces", IEEE Tr. Consumer Electronics, vol. 52-2, p. 559-565, May 2006.