

## Research Article

# 3D Game Content Distributed Adaptation in Heterogeneous Environments

Francisco Morán,<sup>1</sup> Marius Preda,<sup>2</sup> Gauthier Lafruit,<sup>3</sup> Paulo Villegas,<sup>4</sup> and Robert-Paul Berretty<sup>5</sup>

<sup>1</sup> Grupo de Tratamiento de Imágenes, Universidad Politécnica de Madrid, 28040 Madrid, Spain

<sup>2</sup> Département ARTEMIS, Institut National des Télécommunications, 91011 Évry, France

<sup>3</sup> DESICS, Interuniversitair Micro Electronica Centrum, 3001 Leuven, Belgium

<sup>4</sup> Telefónica Investigación y Desarrollo, 47151 Boecillo, Spain

<sup>5</sup> Philips Research, 5656 AE Eindhoven, The Netherlands

Received 31 August 2006; Revised 9 January 2007; Accepted 5 July 2007

Recommended by Yap-Peng Tan

Most current multiplayer 3D games can only be played on a single dedicated platform (a particular computer, console, or cell phone), requiring specifically designed content and communication over a predefined network. Below we show how, by using signal processing techniques such as multiresolution representation and scalable coding for all the components of a 3D graphics object (geometry, texture, and animation), we enable online dynamic content adaptation, and thus delivery of the same content over heterogeneous networks to terminals with very different profiles, and its rendering on them. We present quantitative results demonstrating how the best displayed quality versus computational complexity versus bandwidth tradeoffs have been achieved, given the distributed resources available over the end-to-end content delivery chain. Additionally, we use state-of-the-art, standardised content representation and compression formats (MPEG-4 AFX, JPEG 2000, XML), enabling deployment over existing infrastructure, while keeping hooks to well-established practices in the game industry.

Copyright © 2007 Francisco Morán et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

## 1. INTRODUCTION

OLGA ([www.ist-olga.org](http://www.ist-olga.org)) is the short name of a Specific Targeted REsearch Project (STREP) partially funded, from April 2004 to September 2006, by the European Commission under the Information Society Technologies priority of its Sixth Framework Programme. Its full name comes from its aim: to develop “a unified scalable framework for On-Line GAMing.” The core of the research carried within OLGA, and presented in this paper, was on the challenging topic of 3D game content distributed adaptation in heterogeneous environments. Multiplatform online gaming is an excellent scenario to prove the potential benefits of 4D (animated 3D) content adaptation in the current multimedia world, which still suffers from “platform-centricness” and craves for “user-centricness” and interoperability, despite technological progress over recent years. Indeed, the terminal and network heterogeneity characterising online games make them the perfect example of a completely platform-centric multimedia application:

- (i) game developers must tailor content to specific terminal/network combinations set as a priori targets, and cannot provide the adequate quality for a new such combination without substantial effort;
- (ii) terminal builders and network operators want to diversify their platform characteristics while still being able to allow for game playing;
- (iii) most of all, end-users want to roam attractive games in different usage contexts, inside and outside the home, without being trapped into a single terminal/network configuration.

To overcome platform-centricness and move towards user-centricness, we decided to develop a role-playing game (RPG) [1] named GOAL that would yield similar user experiences on MS Windows-based personal computers (PCs) and on cell phones (CPs) running Symbian OS. Thanks to the OLGA framework, it would be—and in fact is—possible to automatically adapt and render the same textured 4D content at wildly different qualities and frame rates, according



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

to each particular network/terminal profile, as suggested by Figure 1.

We also decided to segregate, from the necessary game logic network, a novel content delivery network which would enable our framework to automatically adapt content in a distributed way. Furthermore, this distributed content delivery network would let players publish their own content and, to this end, we created content authoring tools meant to be used not only by game designers but also by end users. The delivery of content over different networks and its distributed adaptation imposed three basic requirements:

- (i) the volume of data required by the geometry, textures, and animation of 4D models is usually huge, so some form of content *compression* was a must;
- (ii) to ensure interoperability, *de jure* international standards such as MPEG-4 Animated Framework eXtension (AFX) [2], JPEG 2000 [3], and XML [4] would be used, and improved if possible;
- (iii) as content adaptation processes were to run on the same PCs of some of the players, the *processing load* needed by the content adaptation tasks had to be kept to a minimum.

Section 2 elaborates on the core of our research: how scalable coding can be exploited to adapt the different kinds of content data (3D geometry, textures, and animation) in a specific terminal, while achieving excellent quality versus bit-rate versus memory versus execution time tradeoffs.

The rest of our research results highlighted in Sections 3 and 4; the former explains how separating the game logic and content delivery networks allows for a high degree of scalability against the number of clients, and the latter gives some details on the 3D rendering on heterogeneous platforms, stressing our achievements related to auto-stereoscopic displays. Finally, Section 5 concludes our presentation.

## 2. CROSS-PLATFORM AND CROSS-NETWORK 4D CONTENT ADAPTATION

This section presents the core of the research carried out within the OLGA project, which focussed on textured 4D content adaptation for heterogeneous environments (platforms/terminals and networks) based on international standards. The subsections 2.1 to 2.3 below describe how, for each kind of data (3D geometry, 2D textures, animation), different tradeoffs can be achieved between the quality of the decoded content versus the required memory footprint and execution time (which are clearly platform-related aspects) versus the bit-rate (which is mostly a network-related one).

### 2.1. 3D geometry data











Our research in the field of multiresolution coding of static 3D shapes targeted methods more suitable for resource-limited devices than the “WaveSurf” tool in MPEG-4 AFX [2], based on modelling first a given 3D shape (e.g., an arbitrary connectivity mesh) as a wavelet subdivision surface (WSS), and then coding it thanks to the set partitioning in hierarchical trees (SPIHT) technique [5].

Two different types of scalability can be sought in 3D geometry coding, as in the case of image coding: signal to noise ratio (SNR) scalability gives the possibility of decoding a 3D model (or image) with different degrees of fidelity (reconstruction error), whereas spatial scalability allows to decode it with different spatial resolutions, that is, number of vertices/facets (or pixels). For a decade already, SPIHT has been the reference for other scalable coding techniques based on the wavelet transform. It was originally designed to code scalar wavelet coefficients, but has been extended to handle 3D coefficients, such as the ones resulting from RGB images or 3D surfaces modelled thanks to WSSs [6–8].

WSSs are a powerful multiresolution representation paradigm for 3D shapes but the problem of SPIHT is that, although its bit-streams are SNR scalable, they are not spatially scalable. SPIHT bit-streams cannot be easily parsed according to a given maximum resolution (i.e., number of pixels or triangles) or level of detail (LOD, i.e., generation of the subdivision process) tolerated by the decoder, and there is little point in encoding a 3D mesh with thousands of triangles if the CP that must render it can barely handle hundreds, and even less if, anyway, nobody will be able to tell the difference between a 100-triangle mesh and a 1000-triangle one when rendered on a screen of  $200 \times 200$  pixels! Furthermore, from the memory viewpoint (as opposed to the rendering one), having an 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 exhaust 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 outcome of our research is the progressive lower trees of wavelets (PLTW) technique [9], whose main novelty is that the resulting bit-stream does not impose on the less powerful decoders the need of building unnecessary detail trees. With PLTW, the set of wavelet coefficients is also hierarchi-

TABLE 1: Progressive reconstruction of the bunny model from a PLTW bit-stream.

Bit-stream read per LOD	LOD 0: base mesh	LOD 1: one subdivision	LOD 2: two subdivisions	LOD 3: three subdivisions
0%				
50%				
100%				

cally traversed, but coded on a per-LOD basis, thus yielding a bit-stream with “local SNR scalability” and, at the same time, “global spatial scalability.” The decoder first receives all the coefficients corresponding to an LOD and, only when it has finished reading them, it proceeds (if it has enough resources) with those from the next. Nevertheless, thanks to bit-plane encoding, the first received bits from each LOD are the ones contributing more to lower the reconstruction error, while bits from negligible coefficients arrive last. Table 1 shows renderings of the bunny model at different stages of the decoding process. Once the base mesh (LOD 0) is received, it is subdivided once and LOD 1 is progressively reconstructed. When all coefficients of LOD 1 have been decoded, the mesh is subdivided again and details in LOD 2 are processed. LOD 3 and forthcoming levels are sequentially decoded until the whole bit-stream is read.

Figure 2 shows the rate-distortion (PSNR as a function of the number of bits per vertex) curves obtained for two typical 3D models by our PLTW coder, which includes arithmetic coding (AC) as a final step, and an a version of the SPIHT algorithm with AC. In the case of SPIHT, the bits from each LOD have been individually plotted: LODs 1 and 2 are quickly reconstructed because their details are the ones contributing the most to lower the reconstruction error. It would seem clever to cut the stream or stop decoding after some point (e.g., 0.75 b/v for LOD 1 or 1.5 b/v for LOD 2) if those two coarsest LODs are enough, or the only manageable ones, since the bits to come will hardly increase the PSNR. However, even in those cases, the decoder needs to build the whole detail tree to be able to follow the branching of the SPIHT algorithm. On the contrary, the PLTW decoder is able to stop decoding exactly at the desired LOD without allocat-

ing extra resources for further LODs—and even with a lower reconstruction error!

Figure 3 plots the rate distortion curves for the PLTW coder, the same AC version of SPIHT as above (overall compression shown in one curve), and the “WaveSurf” tool of MPEG-4, which also uses SPIHT, but without AC. Except at very low rates, where the 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 quantisation 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.

WSSs permit to code the shape of a 3D model in a multiresolution 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. Figure 4 shows that 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 [10, 11]. 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 of “WaveSurf” with the PLTW

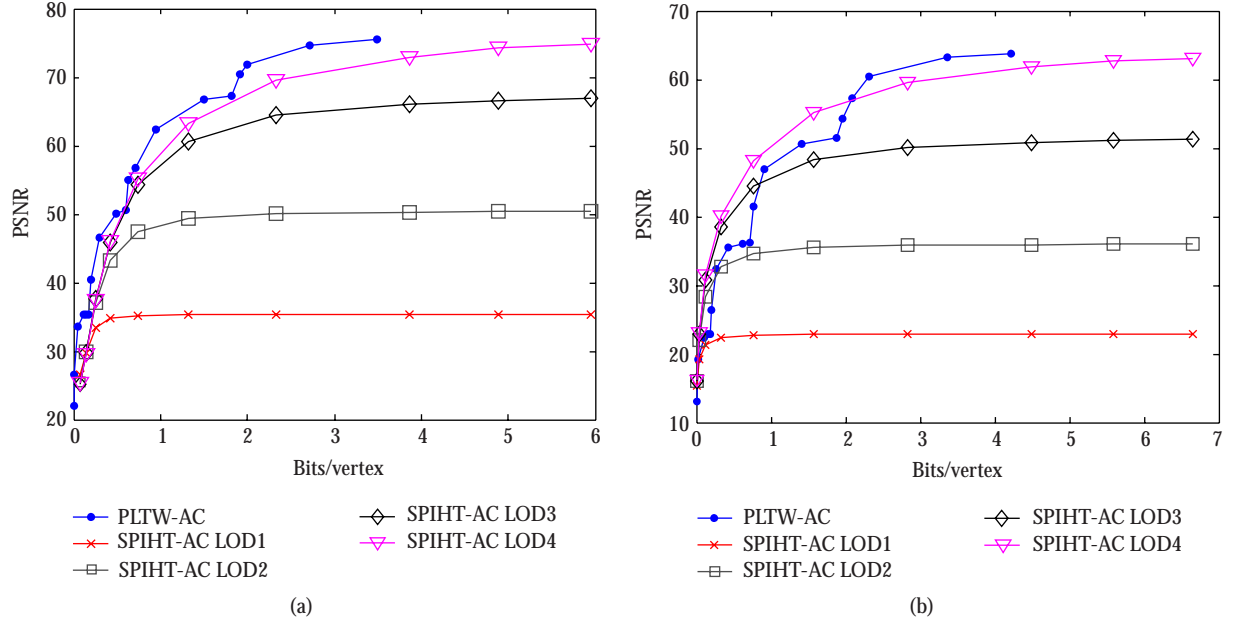


FIGURE 2: PLTW versus SPIHT for the Max Planck (a) and bunny (b) models.

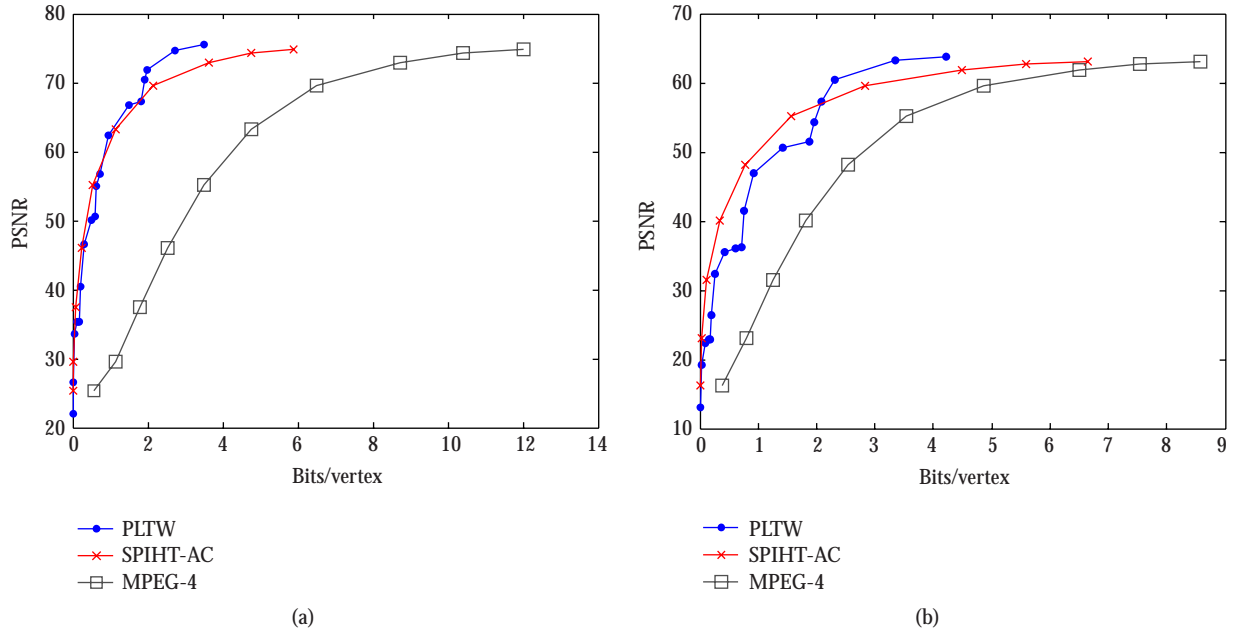


FIGURE 3: PLTW versus SPIHT and MPEG-4's "WaveSurf" for the Max Planck and bunny models.

technique, we have introduced some add-ons to enable a low-complexity, yet efficient fine-grained quality/execution time tradeoff in execution time control, as shown by the upper curves of Figures 4 and 6.

To achieve this target, the "WaveSurf" mesh regions are progressively decoded in a continuous LOD fashion, by subdividing only the important regions of the geometry, as shown in Figure 5. The importance and order for subdividing the triangles is given by their impact on the error to the target mesh, that is, the triangles that decrease this er-

ror the most are subdivided first. We detect importance with a heuristic [12] for which values smaller or larger than a threshold parameter  $h_t$  are, respectively, detected as important or unimportant. Nonuniform subdivision of a WSS does not necessarily create all four children of a triangle, but first checks the importance of every potential child for a specific value of  $h_t$  before it is added. In this way, additional triangles are only introduced (and hence execution time increased) when they really contribute to an improved visual quality of the mesh. Of course, one must then worry about the "cracks"



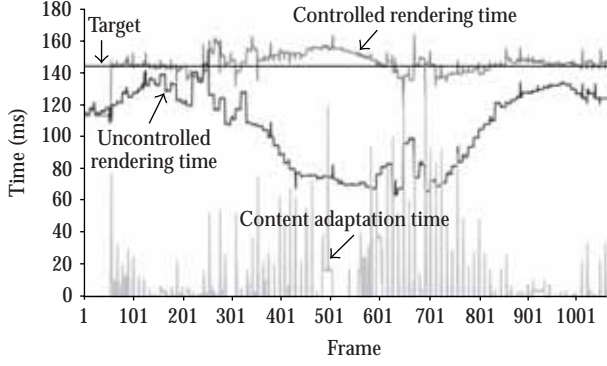


FIGURE 4: Content adaptation for execution time control.

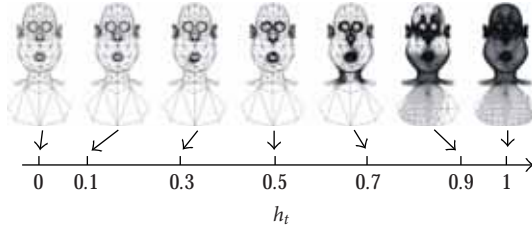
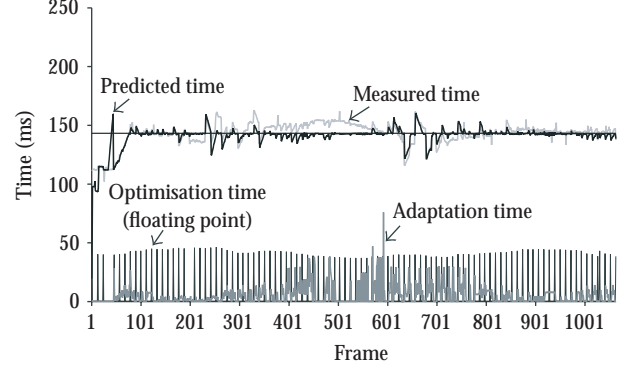


FIGURE 5: Continuous LOD with adaptively subdivided WSSs.

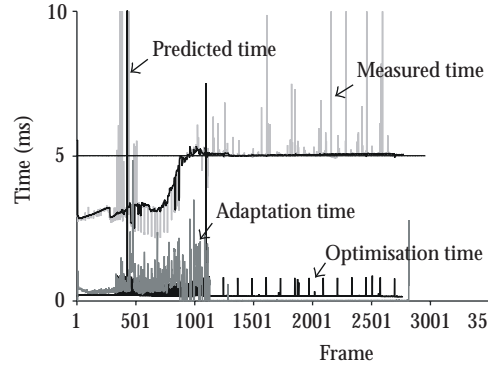
that may appear when rendering the mesh, but this problem can be easily solved [7, 13].

These nonuniformly subdivided WSS 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 techniques using LOD-based moving windows [10], 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, as shown in Figure 6 for two different terminals: a high-end PC and a low-end CP [11].

To 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 [13] and hardware [14] rendering pipelines, 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 rasterisation); 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 offline 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  $c_\alpha$  ( $\alpha \in \{T, F, S\}$ ) through multilinear regression analysis. A mean error of only 5% between estimated and measured execution time has been observed in the case of the software rendering pipeline [13].



(a) Software: embedded device



(b) DirectX: PC-Geforce Fx5900t

FIGURE 6: Controlled execution time on two different platforms for a 3D scene walkthrough with a moving character.

## 2.2. Combined 2D textures + 3D geometry data

In order to choose a coding tool and format for textures, we first carried a comparative study, with respect to the considered criteria and desired functionalities, between the classical DCT-based solution, JPEG, and two wavelet-based, and also already standardised solutions: JPEG 2000 [3] and MPEG-4's native tool for still images, VTC (Visual Texture Coding) [15]. We chose JPEG 2000, which is now supported inside MPEG-4 as a format for textures thanks to a proposal of ours.

Besides the execution time variation with the platform and content parameters [13], we also observed the linearity of the cost with the object parameters 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 decreasing JPEG 2000 texture LOD (with negative slope  $-m_1$ ) and decreasing object mesh LOD (with negative slope  $-m_2$ ). Figure 7 illustrates this trend for two OLGA objects.

Small file sizes  $s$  with large  $m_1$  and  $m_2$  correspond to small bit-rates that decrease very rapidly with decreasing LOD: those 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

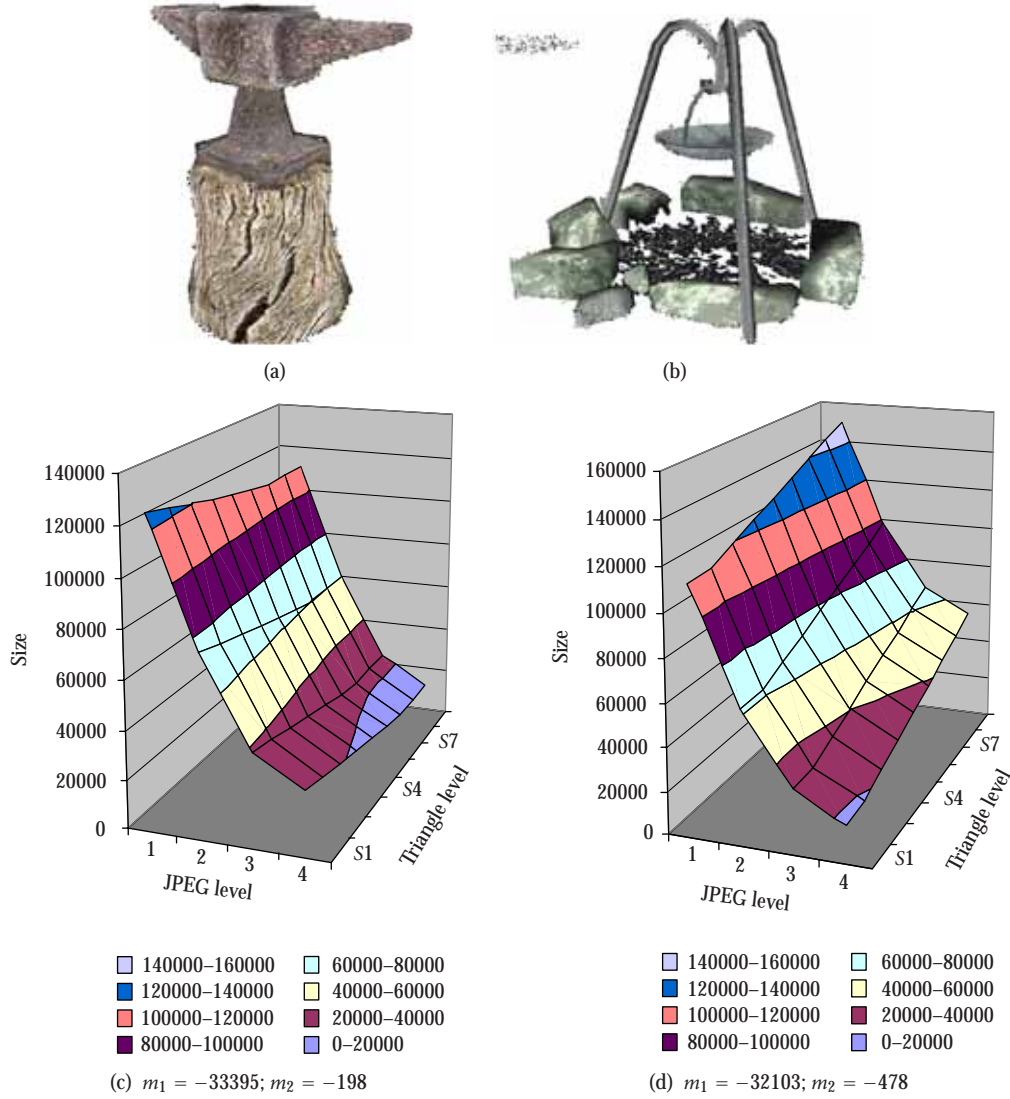


FIGURE 7: Bilinear dependency of MPEG-4 file size on 2D texture and 3D mesh LODs.

LOD, hence representing hardly any chance of downscaling 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 PLTW and BBA tools (see below), a global quality versus bit-rate versus execution time control can be obtained over all objects. The precise details of this intelligent global adaptation are beyond the scope of this paper, since it is mainly related in finding heuristics for approximately solving an NP-hard knapsack problem: the interested reader is referred to [16, 17] for a framework of 3D interobject adaptation using some tabular characteristics of each object.

### 2.3. Animation data

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 standardised an approach for compression of generic interpolated data [18], 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, ISO/IEC published in 1999 [15] and 2000 [19], under the umbrella of the MPEG-4 specifications, is a set of tools named face and body animation (FBA) [20] allowing compression at very low bit-

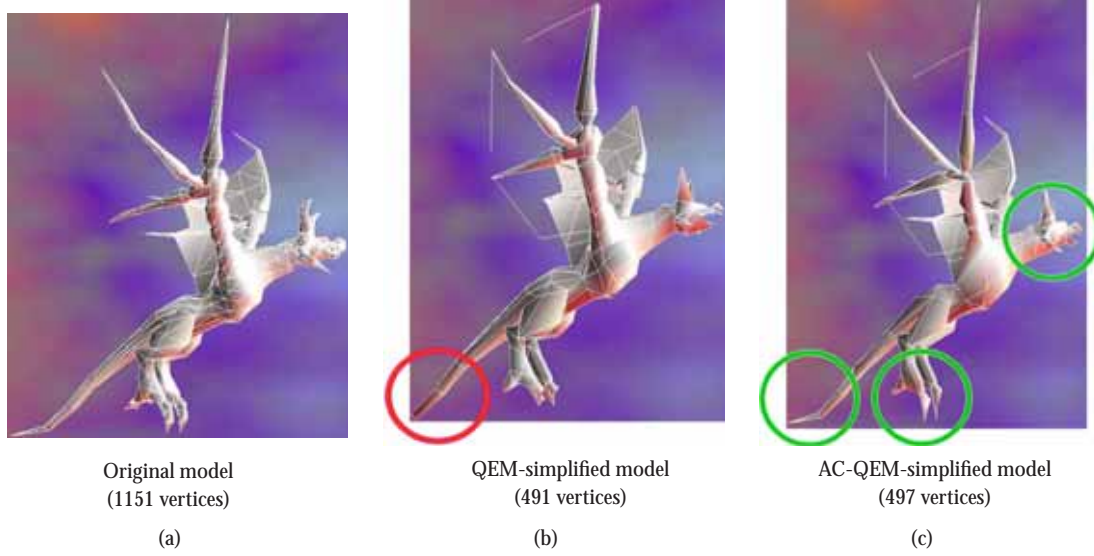
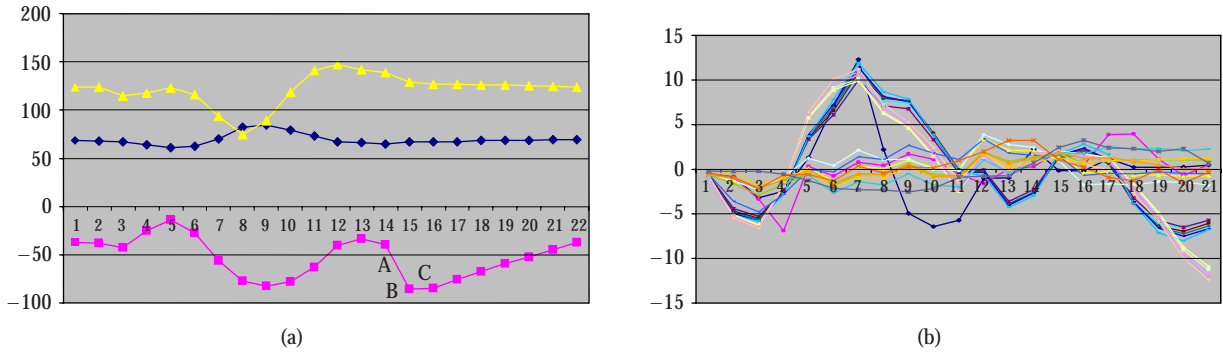


FIGURE 8: AC-QEM versus QEM: qualitative results for the dragon model.

FIGURE 9: Movement along the  $x$  (■),  $y$  (◆), and  $z$  (▲) axes of the centre of a single bone (a) and all extreme bones (b).

rates. The limitations of FBA consist mainly in the rigid definition of the avatar and the difficulty to set up the proposed deformation model. Some other methods reported in the literature are quantisation of the motion type [21], data transmission scalability by exploiting the 3D scene structure [22], and quantisation to achieve data compression and incorporate intelligent exploitation of the hierarchical structure of the human skeletal model [23].

At the time the OLGA project started, we were [24] in the final stage of standardising BBA, an extension of FBA within MPEG-4 AFX [2]. 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 behaviour [25] and animation frame reduction.

The dynamic behaviour was expressed as constraints used to parameterise the well-known mesh simplification quadric error metrics (QEM) technique [26]. 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 8 shows the comparative results of an animated model simplification for the developed approach, called animation-constrained QEM (AC-QEM), versus plain QEM.

Decoding and rendering animation data on small memory devices such as CPs require severe server-side compression. To decrease even more the size of animation data, we implemented a frame reduction algorithm: instead of transmitting all frames, we have the server transmit some key frames only, and let the decoder guess intermediate frames by linear interpolation (NB: MPEG-4 supports nonuniform temporal interpolation by indicating at each key frame the number of intermediate frames to be computed).

Given an original animation sequence of  $n$  frames, to obtain a simplified sequence with  $m$  frames ( $m < n$ ) approximating best the original curve, the area between the original

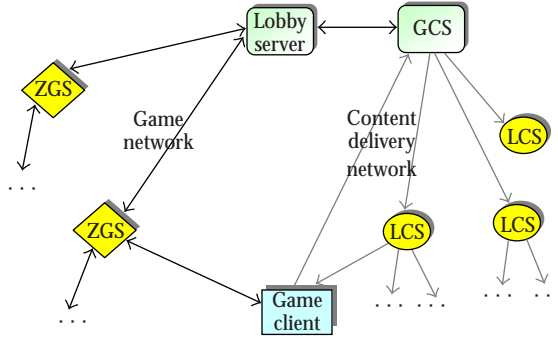


FIGURE 10: Overall network architecture.

curve and the one reconstructed by linear interpolation must be minimised. For instance, Figure 9(a), showing the movement during 22 frames of the  $x$ ,  $y$ , and  $z$  coordinates of the centre of a bone, illustrates how removing frames no. 4 or no. 19 is less critical, from a distortion viewpoint, than removing frames no. 3 or no. 8.

Considering this condition for all bones, or even for the subset of extreme bones, as shown in Figure 9(b), the optimisation problem becomes difficult to solve. To overcome the complexity, we adopted an incremental approach.

- (i) We first compute for each extreme bone, frame, and coordinate, the area of the triangle defined by the original curve and the one reconstructed by linear interpolation. In the case of the bone of Figure 9(a), its  $x$  coordinate for frame no. 15 (marked as B) would be reconstructed (erroneously) by linear interpolation between its values in frames no. 14 (A) and no. 16 (C), so the area of triangle ABC is a measure of the error caused by omitting frame no. 15.
- (ii) Then, for each frame, we sum these all the error areas for all extreme bones and coordinates. The minimum of the sums indicates the frame that has to be removed. We iterate the algorithm until only  $m$  frames remain, and generate a new BBA stream by encoding those  $m$  frames, indicating for each the number of intermediate frames to be obtained by linear interpolation on the terminal.

The advantage of this incremental approach where frames are removed one by one is the fine granularity of the file size: in a time-variant environment of the network capabilities, it is possible to dynamically adapt the size of the animation stream to the changing constraints of the network.

### 3. CROSS-NETWORK DISTRIBUTED CONTENT DELIVERY

#### 3.1. Overall network architecture

The implemented network architecture follows a dual design. There are two different subnetworks within the system, shown in Figure 10.

- (i) The game network (GN) holds the game logic, keeping synchronisation among its nodes and therefore en-

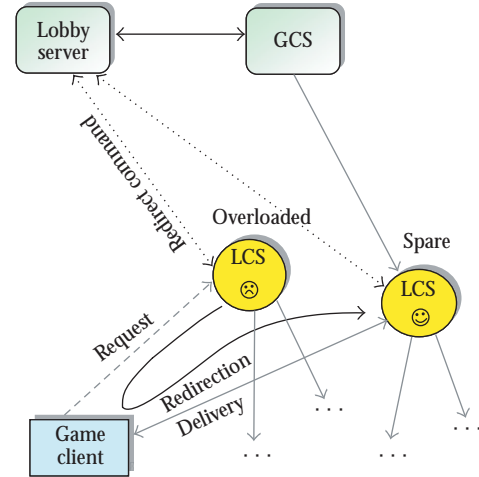


FIGURE 11: Dynamic load balancing mechanism.

abling a multiplayer online game. We built it, as a mere support service for our system, with off-the-shelf components, so the implemented game engine can cope with heterogeneous clients and networks, using standard protocols (HTTP, XML-RPC, etc.) to help interoperability.

We chose a turn-based role-playing game (RPG) as a test bed, instead of a faster-paced game genre, to soften the requirements on the GN; nevertheless, we added some basic tools such as dead reckoning [27, 28] and simple latency equalisation to ensure that clients had a comparable user experience. User validation showed indeed that there was no statistically significant difference between PC and CP players with respect to game experience, and network delay did not negatively affect the players' impression, provided it stayed within certain bounds.

The GN architecture is distributed in the sense that different matches are hosted by separate servers, called zone game servers (ZGSs). ZGSs are run by client PCs: in every match, one of the clients acts as the game logic server for all others (and itself, of course). A dynamic procedure governed by the central lobby server (LS) decides upon creation of each match which client will host its ZGS. The LS monitors the ZGS while the match is going on and, if it detects that the ZGS fails (or disconnects from the game, perhaps because its owner simply decides to switch it off), it starts a replacement ZGS on another PC and transfers all clients to it. In such a case, the game state is (mostly) preserved thanks to the backups that are sent periodically from the ZGS to the LS.

- (ii) The content delivery network (CDN) is the specialised subsystem that we developed to enable live 4D content update while playing, and to perform dynamic adaptation of that content to terminals in a distributed fashion. It is formed by a global content server (GCS), the single point of upload for new content, and a number of adaptation and delivery nodes called local content



servers (LCSs) that, analogously to the ZGS, are also hosted by game clients.

Both networks, GN and CDN, meet at their edges, as the LS coordinates both and game clients also connect to both, and it is not unusual that a single client PC be a node in both networks, since it can host both a ZGS and an LCS. However, from a logical point of view, they are different entities.

### 3.2. Content delivery network

Sending or updating game content (i.e., objects to be rendered) over the network is not a frequently used option, although multiplayer online games pushing content through the network instead of locally storing all data do exist [29]. However, most of these games reduce a priori the transmission bandwidth by subdividing the world in subworlds (3D tiles) and referencing prestored 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 peer-to-peer (P2P) model was chosen. Distributing content across networks is one area in which P2P technologies have experienced an enormous boom recently [30]. In our case, since the main focus of our work was on-the-fly content adaptation, our system is a hybrid of a content delivery network and a distributed computing system for which, instead of a generic and heavyweight approach such as grid computing [31], we chose a more specialised distributed P2P computing model. The background idea comes from several large-scale experiments done on Internet computing, the most famous being probably SETI@home [32].

Our target is then to use residual computing power from the client nodes, which must be kept free enough to perform their own individual tasks, notably playing the game. Some studies have been done on user acceptability of the use of system resources by external processes [33]; our objective has been that the adaptation tasks never exceed a given threshold on system load. This is enforced by the load balancing procedure described below.

The final key features of the CDN are the following.

- (i) 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.
- (ii) 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 a set of simplification tools [34], and the LCSs cache the result of simplifications to save processing effort.
- (iii) 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, as explained in Section 3.

- (iv) 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 CDN, some scalability is inherent to it: it is likely that new game clients entering will also bring new servers, thus levelling 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 must be provided.

Each LCS serves a number of game clients, following an arrangement that can evolve to accommodate changes in network and client conditions. On initialisation, a game client connects to the LS and sends a request for joining a game. As side information, it also sends its node characteristics, that is, computing power and network bandwidth. The LS logs in each game client and assigns it a ZGS, in charge of delivering all the game logic to the client, and an LCS, which will deliver all the game content elements to it. In parallel, and depending on node characteristics, the LS may tell the client to start an LCS, which from then on listens to requests for adapted content (it may also tell the client to start an inactive LCS, keeping it for future needs).

The delivery process involves continuous interaction between the client and its two assigned servers: the client interacts with the game world as it is given by the ZGS; whenever the ZGS delivers indication of a new content element, the client contacts its LCS and requests that item, which is then downloaded and rendered. Each request consists of an object identifier and a list of quality parameters. The LCS proceeds to optimise the locally cached content (or download it, on cache misses, from the GCS) according to those parameters by using the developed content adaptation tools, and generates a bit-stream encapsulating the optimised content, in a format suitable for the client. All meshes, animations, and so forth, that are used in a game, are retrieved from the LCS.

After the original initialisation, clients do not contact the LS again: all interaction is done through their LCS, including reassignments to another LCS. Given that network delays in reassignments may be significant since LCSs may be topologically separated, job migration is minimised by establishing a long-term procedure: clients are assigned an LCS upon entering the network, and they continue using the same LCS until they are reassigned. The load balancing procedure, represented in Figure 11, is as follows.

- (i) A threshold is put on the maximum CPU load and bandwidth of the LCS, depending on the device characteristics and connectivity. Obviously, that threshold is set well below 100% (load), since the LCS machine is after all mostly a game client, and we do not want the LCS adaptation processes to hamper the game experience.

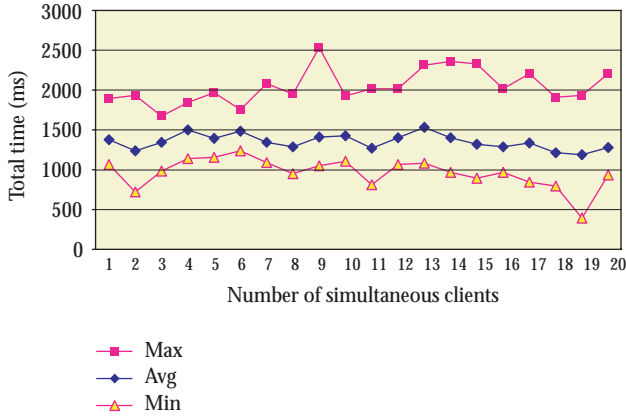


FIGURE 12: LCS total serve time versus number of clients.

- (ii) The LS polls CPU load and bandwidth usage of the LCS at regular intervals, to monitor its status. To avoid exceeding the threshold between polls, prediction over the last received data is used so that short-term future conditions are anticipated. Simple linear prediction has been initially chosen, but other schemes are also possible [35, 36].
- (iii) When the threshold is expected to be shortly reached, the LS chooses a replacement from the pool of available LCSs, and sends a message to the loaded LCS to redirect all future adaptation requests to the replacement; game clients change their assigned LCS as they receive such redirections. If the LCS load later goes below a safety value, the LS tells it to start accepting adaptation tasks again.

This procedure could be classified as “local decision global migration” [37]: the decision to redirect a job is taken based only on the state of the affected node (local decision), but once decided to be transferred, the job can travel anywhere in the CDN (global migration).

The load produced by adaptation tasks may depend on various factors: the content to adapt, the adaptation parameters (e.g., the simplification level), or the number of concurrent adaptations. The resulting serve time is, in general, the sum of the time needed for adaptation plus the time for delivery; if the LCS has already the adapted content, only the latter time will count; if the LCS does not have the content at all, the time needed to fetch it from the GCS must be added as well to the sum above.

Figure 12 shows the total serve time per client on an LCS that is busy doing a number of adaptation tasks, as a function of the number of simultaneous adaptations, using a medium-powered machine with a broadband connection. To eliminate the dependency from the content, all simplification tasks were made over the same file (always ensuring that adaptation was forced, instead of using an adapted version in the cache). As the graph shows, the system scales up quite well, since the time to adapt and deliver content is independent from the number of simultaneous adaptations (provided the load does not go beyond a congestion limit). Most of that time is spent in adaptation: simplification of the con-

tent takes on average 85% of the total time, while only 15% is spent in delivery (obviously, if the adapted content were in cache, the serve time would improve considerably), since we have favoured lower bandwidth usage.

LCS load and bandwidth usage values were also measured against the number of simultaneous clients being served. Variations using different content files show different practical limits in the number of simultaneous clients, but the general shape stays the same. Also, it has been found that there is no significant impact of the simplification level on total adaptation time, for a given content file. Since in general it is very difficult to know beforehand the pattern of adaptations that an LCS is going to receive (the requested content depends from the behaviour of the client within the game), load prediction and balancing is important to be able to keep LCSs under control.

A similar procedure to that for load balancing is used for fault tolerance. In case a client does not receive timely responses from its LCS, it will contact the LS and ask for a replacement. The LS will assess the situation (due to polling it would probably have already detected the problem) and assign the best available LCS to the client.

We have also investigated a number of variants on the network architecture to improve efficiency and better adapt to different needs in content distribution. Some of these alternatives are the following.

- (i) Enabling peer cache lookups, through a “hit or nothing” procedure: the peer only delivers the content if it is in its cache and already adapted as needed; neither adaptations nor upstream GCS fetches are resolved. This is similar in concept to the sibling cache process in the Internet Cache Protocol [38].
- (ii) 2-level hierarchy: a new level between the GCS and the LCS, with nodes called content aggregators (CAs), acting as a higher level cache, gathering both unadapted content upstream (from the GCS) and adapted content downstream (from the LCS). If the LCS receives unadapted content, as soon as it finishes the adaptation it delivers it to both the original requester and the CA, thus growing the CA cache.

Though the topology of the network increases in complexity with the latter versions, the protocol itself remains quite simple and the procedure followed by each node is straightforward, thus ensuring general simplicity in each node.

Finally, game interactivity can be further improved by enhancing the system with techniques such as automatic coarse adaptation and delivery of items (to create fast previews), background anticipative prefetching of items as suggested by the game engine or priority queues in adaptation servers.

## 4. CROSS-PLATFORM 3D RENDERING

### 4.1. Software platforms

The OLGA technology supports a variety of terminals, 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.

- (i) 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.
- (ii) Besides, we have developed a stand-alone MPEG-4 player to visualise textured 4D content on both PCs and CPs. We selected a small number of the scene graph nodes defined in the MPEG-4 Systems specification [39], which is enough to represent static and animated textured 3D objects. Then, starting from GPAC [40], an open source decoder of MPEG-4 Binary Format for Scene (BIFS), 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 optimised the initial BBA decoder we had developed for PC and also ported it to Symbian OS v8 for the selected CP (Nokia 6630). Finally, we developed the rendering layer by using DirectX 9 for PC and OpenGL ES for CP. Figure 13 shows some models loaded in the PC and CP versions of the 3D Graphics MPEG-4 player.

#### 4.2. 3D displays

Special attention has given to the final rendering of 3D gaming content. Nowadays, new terminal and display developments drive new applications such as real 3D viewing. It has been demonstrated how to optimally design lenticular sheets to turn a flat, 2D matrix display into an auto-stereoscopic multiview display [41]. The resulting displays can present the viewer with different images from various slightly different viewing angles. A format highly suitable for content transmission for such multiview displays is video enriched with depth information [42, 43]. 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 [44].

To make the game experience more immersive [45], we have endowed some terminals with such autostereoscopic 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 [44]. In the GOAL terminals, we have 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 multiview 3D, as suggested by Figure 14—which, of course, cannot convey the real 3D experience when printed on 2D paper!

## 5. CONCLUSIONS

Today's multiplayer 4D 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 consortium opted for a completely different paradigm: thanks to scalable coding of the 3D geometry, texture, and animation data, gaming content is automatically adapted to heterogeneous platforms and networks, and the processing load distributed among the resources available in a P2P architecture. Indeed, OLGA's 4D 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 new specialised 4D content compression tools are provided to end users as well as to game designers, the players can publish their own content, which then becomes part of the persistent world, and benefits from OLGA's standardised 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. These 4D content compression tools do not impose constraints on content complexity: game developers and players are free in their creativity, and the tools adapt to any circumstances—not the other way around, as is usually the case.

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



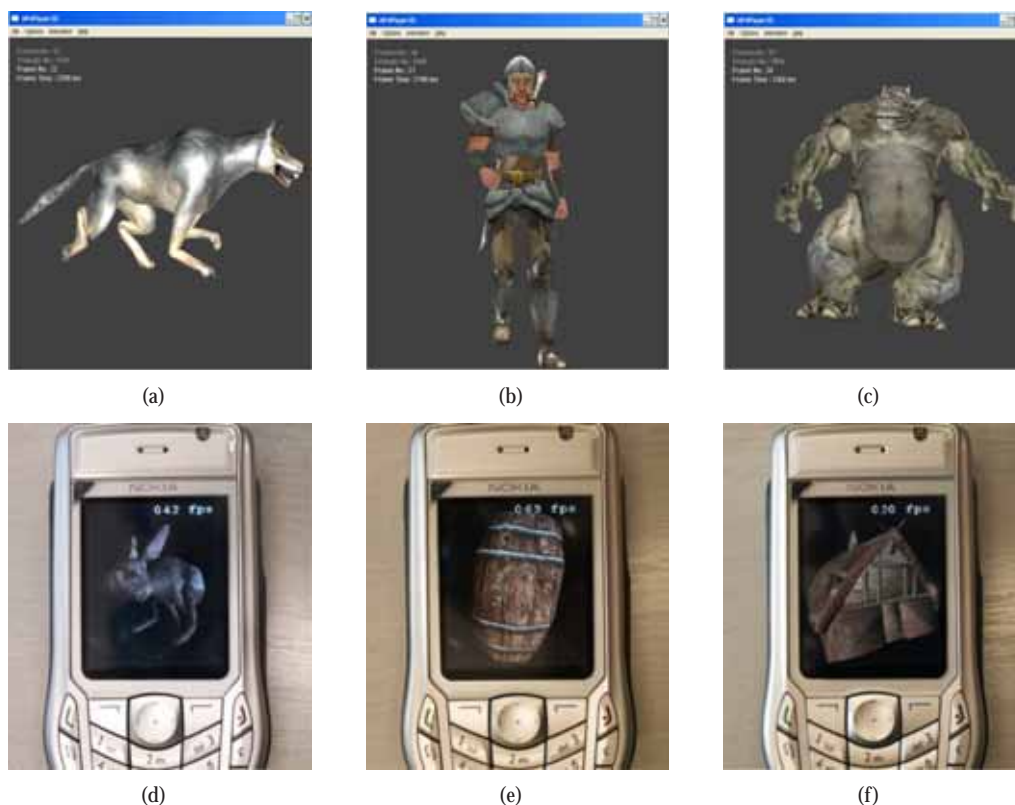


FIGURE 13: Some OLGA models loaded in the PC (top) and CP (bottom) versions of the 3D graphics MPEG-4 player.



FIGURE 14: Screen shot of GOAL: image and depth buffers (a), and actual image on autostereoscopic 3D display (b).

## REFERENCES

- [1] T. H. Apperley, "Genre and game studies: toward a critical approach to video game genres," *Simulation & Gaming*, vol. 37, no. 1, pp. 6–23, 2006.
- [2] ISO/IEC JTC1/SC29/WG11, "Standard 14496-16," a.k.a. "MPEG-4 Part 16: Animation Framework eXtension (AFX)", ISO, 2004.
- [3] 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", ISO, 2004.
- [4] XML (eXtensible Markup Language) Core Working Group, "XML 1.0 (4th ed.)," W3C (World Wide Web Consortium), <http://www.w3.org/TR/2006/REC-xml-20060816/>, 2006.
- [5] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.
- [6] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 271–278, ACM, New Orleans, La, USA, July 2000.
- [7] F. Morán, *Modelado jerárquico de objetos 3D con superficies de subdivisión*, Ph.D. thesis, Universidad Politécnica de Madrid, Madrid, Spain, 2001, <http://www.gti.ssr.upm.es/~fmb/pub/PhD.pdf.gz>.
- [8] F. Morán and N. García, "Comparison of wavelet-based three-dimensional model coding techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 937–949, 2004.
- [9] M. Avilés, F. Morán, and N. García, "Progressive lower trees of wavelet coefficients: efficient spatial and SNR scalable coding of 3D models," in *Proceedings of the 6th Pacific Rim Conference on Multimedia (PCM '05)*, vol. 3767 of *Lecture Notes in Computer Science*, pp. 61–72, Springer, Jeju Island, Korea, November 2005.
- [10] N. Tack, G. Lafruit, F. Catthoor, and R. Lauwereins, "Eliminating CPU overhead for on-the-fly content adaptation with MPEG-4 wavelet subdivision surfaces," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 559–565, 2006.
- [11] K. Tack, G. Lafruit, F. Catthoor, and R. Lauwereins, "Platform independent optimisation of multi-resolution 3D content to enable universal media access," *The Visual Computer*, vol. 22, no. 8, pp. 577–590, 2006.
- [12] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: multiresolution adaptive parameterization of surfaces," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pp. 95–104, ACM, Orlando, Fla, USA, July 1998.
- [13] N. Tack, F. Morán, G. Lafruit, and R. Lauwereins, "3D graphics rendering time modeling and control for mobile terminals," in *Proceedings of the 9th International Conference on 3D Web Technology*, pp. 109–117, ACM, Monterey, Calif, USA, April 2004.
- [14] M. Wimmer and P. Wonka, "Rendering time estimation for real-time rendering," in *Proceedings of the 14th Eurographics workshop on Rendering*, pp. 118–129, Leuven, Belgium, June 2003.
- [15] ISO/IEC JTC1/SC29/WG11, "Standard 14496-2," a.k.a. "MPEG-4 Part 2: Visual," ISO, 1999.
- [16] J. Bormans, N. P. Ngoc, G. Deconinck, and G. Lafruit, "Terminal QoS: advanced resource management for cost-effective multimedia appliances in dynamic contexts," in *Ambient Intelligence: Impact on Embedded System Design*, pp. 183–201, Kluwer Academic Publishers, Norwell, Mass, USA, 2003.
- [17] P. N. Ngoc, G. Lafruit, J. Y. Mignolet, S. Vernalde, G. Deconinck, and R. Lauwereins, "A framework for mapping scalable networked applications on run-time reconfigurable platforms," in *Proceedings of International Conference on Multimedia and Expo (ICME '03)*, vol. 1, pp. 469–472, Baltimore, Md, USA, July 2003.
- [18] E. S. Jang, J. D. K. Kim, S. Y. Jung, M.-J. Han, S. O. Woo, and S.-J. Lee, "Interpolator data compression for MPEG-4 animation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 989–1008, 2004.
- [19] ISO/IEC JTC1/SC29/WG11, "Standard 14496-2/AMD1," a.k.a. "MPEG-4 Part 2: Visual, Amendment 1: Visual extensions," ISO, 2000.
- [20] M. Preda and F. Prêteux, "Critic review on MPEG-4 face and body animation," in *Proceedings of the International Conference on Image Processing (ICIP '02)*, vol. 3, pp. 505–508, Rochester, NY, USA, September 2002.
- [21] M. Endo, T. Yasuda, and S. Yokoi, "A distributed multiuser virtual space system," *IEEE Computer Graphics and Applications*, vol. 23, no. 1, pp. 50–57, 2003.
- [22] T. Hijiri, K. Nishitani, T. Cornish, T. Naka, and S. Asahara, "Spatial hierarchical compression method for 3D streaming animation," in *Proceedings of the 5th Symposium on Virtual Reality Modeling Language (Web3D-VRML '00)*, pp. 95–101, ACM, Monterey, Calif, USA, February 2000.
- [23] S. Chattopadhyay, S. M. Bhandarkar, and K. Li, "Virtual people & scalable worlds: efficient compression and delivery of stored motion data for avatar animation in resource constrained devices," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '05)*, pp. 235–243, ACM, Monterey, Calif, USA, November 2005.
- [24] M. Preda and F. Prêteux, "Virtual character within MPEG-4 animation framework eXtension," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 975–988, 2004.
- [25] M. Preda, S. Tran, and F. Prêteux, "Adaptation of quadric metric simplification to MPEG-4 animated object," in *Proceedings of the 6th Pacific Rim Conference on Multimedia (PCM '05)*, vol. 3767 of *Lecture Notes in Computer Science*, pp. 49–60, Springer, Jeju Island, Korea, November 2005.
- [26] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pp. 209–216, ACM, Los Angeles, Calif, USA, August 1997.
- [27] IEEE, "Standard 1278.1a for Distributed Interactive Simulation—Application Protocols," IEEE, 1998.
- [28] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly, "Department of defense high level architecture," in *Proceedings of the 29th Conference on Winter Simulation*, pp. 142–149, ACM, Atlanta, Ga, USA, December 1997.
- [29] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of networking in multiplayer computer games," *Electronic Library*, vol. 20, no. 2, pp. 87–97, 2002.
- [30] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.
- [31] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.
- [32] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.
- [33] A. Gupta, B. Lin, and P. A. Dinda, "Measuring and understanding user comfort with resource borrowing," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pp. 214–224, Honolulu, Hawaii, USA, June 2004.
- [34] F. Morán, M. Preda, G. Lafruit, P. Villegas, and R.-P. Berretty, "A content adaptation approach for on-line gaming on var-

- ious networks and terminals," in *Proceedings of International Digital Games Conference (iDiG '06)*, pp. 233–236, Portalegre, Portugal, September 2006.
- [35] P. A. Dinda and D. R. O'Hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, no. 4, pp. 265–280, 2000.
  - [36] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 757–768, 1999.
  - [37] R. Lüling, B. Monien, and F. Rammme, "Load balancing in large networks: a comparative study," in *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, pp. 686–689, Dallas, Tex, USA, December 1991.
  - [38] D. Wessels and K. Claffy, "RFC 2187: Application of Internet Cache Protocol (ICP), version 2," IETF, 1997.
  - [39] ISO/IEC JTC1/SC29/WG11, a.k.a. MPEG (Moving Picture Experts Group): "Standard 14496-1", a.k.a. "MPEG-4 Part 1: Systems", ISO, 1999.
  - [40] J. Le Feuvre, "GPAC," <http://gpac.sourceforge.net/>.
  - [41] C. van Berkel and J. A. Clarke, "Characterization and optimization of 3D-LCD module design," in *Stereoscopic Displays and Virtual Reality Systems IV*, vol. 3012 of *Proceedings of SPIE*, pp. 179–186, San Jose, Calif, USA, February 1997.
  - [42] A. Redert, M. O. de Beeck, C. Fehn, et al., "ATTEST: advanced three-dimensional television system technologies," in *Proceedings of 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT '02)*, pp. 313–319, Padova, Italy, June 2002.
  - [43] C. Fehn, P. Kauff, M. O. de Beeck, et al., "An evolutionary and optimised approach on 3D-TV," in *Proceedings of International Broadcast Conference (IBC '02)*, pp. 357–365, Amsterdam, The Netherlands, September 2002.
  - [44] R.-P. Berretty, F. J. Peters, and G. T. G. Volleberg, "Real time rendering for multiview autostereoscopic displays," in *Stereoscopic Displays and Virtual Reality Systems XIII*, vol. 6055 of *Proceedings of SPIE*, pp. 208–219, San Jose, Calif, USA, January 2006.
  - [45] R. Rajae-Joordens, E. Langendijk, P. Wilinski, and I. Heynderickx, "Added value of a multi-view auto-stereoscopic 3D display in gaming applications," in *Proceedings of the 12th International Display Workshops in Conjunction with Asia Display (IDW/AD '05)*, pp. 1731–1734, Takamatsu, Japan, December 2005.

**Francisco Morán** received the degrees of Telecommunication Engineering and Ph.D. in telecommunication, both from the UPM (Universidad Politécnica de Madrid, Spain), in 1992 and 2001, respectively. Since 1992, he is a Researcher at UPM's GTI (Grupo de Tratamiento de Imágenes-Image Processing Group). In 1997, he became a member of the faculty of UPM's Department of Signals, Systems, and Communications, where he is, since 2002, a full time Associate Professor in the knowledge area of Signal Theory and Communications. His research interests include modelling, coding, transmission, and visualisation of 3D objects. He has been and is actively involved in projects funded by the RACE, ACTS, and IST Programmes of the European Commission, for example, OLGA, a STREP from FP6 that ended successfully in 2006. He also participates actively since 1996 in the standardisation activities from ISO's moving picture experts group (MPEG). He is (co-)editor of several standards, amendments, and corrigenda related to MPEG-4 (formally, ISO/IEC 14496), and he is the Head of the Spanish Delegation of MPEG since 2006.



**Marius Preda** is Associate Professor at the ARTEMIS Department of the GET INT (Groupe des Ecoles des Télécommunications-Institut National des Télécommunications) in Evry, France. He received an Engineering degree in Electronics from University Politehnica of Bucharest (Romania) in 1998, and a Ph.D. degree in mathematics and informatics from Université Paris V-René Descartes (France) in 2001. He started his carrier in Bucharest as a production engineer at Electronica Aplicata and then as a researcher at University Electronica si Telecomunicatii. During his Ph.D. studies, he was R&D Engineer at GET INT ARTEMIS, where he held an R&D Project Manager position in 2003–2004 before becoming an Associate Professor in 2005. His interests include 3D graphics representation and compression, multimedia composition, interactive applications, and multimedia standardization. He is actively involved in the management of various projects at the national and European levels. He is the Chairman of the 3D Graphics subgroup of Moving Picture Experts Group (MPEG) and editor of the standards ISO/IEC 14496 16, Animation Framework eXtension (AFX) and ISO/IEC 14496 25, 3DGCM (3D Graphics Compression Model). He is the initiator of MPEG's 3DGCM project, aiming at a generic 3D graphics architecture model allowing to apply MPEG-4 compression on any scene description formalism.



**Gauthier Lafruit** was a Research Scientist with the Belgian National Foundation for Scientific Research from 1989 to 1994, being mainly active in the area of NMR image acquisition, wavelet image compression, and VLSI implementations for image transmission. From 1995, he was a Research Assistant at the Department of Electronics with the Vrije Universiteit Brussel, Belgium. In 1996, he joined IMEC, where he was first involved as a Senior Scientist in projects for the low-power VLSI implementation of combined JPEG/wavelet compression engines for the European Space Agency. His main current activities are with progressive transmission in still image and video coding, scalability in textured 3D models rendering, and resource monitoring in virtual reality. In this role, he has made decisive contributions to the standardisation of 3D implementation complexity management in MPEG-4. His scientific activities are evolving towards multicamera video coding, transmission, stereo matching, and visual viewpoint interpolation, with as central interest point the well-balanced tradeoff between algorithmic, architectural, and compiler efficiency. The main target is the development of an overall optimised system, both end-to-end, as from the perspective of design effort and run-time efficiency. He is (co)author of over one hundred scientific publications, four patents, and a couple of book chapters and science vulgarization articles.



**Paulo Villegas** holds a Telecommunications Engineer degree from Universidad Politécnica de Madrid. He has been working at Telefónica I + D since 1992, first in Madrid and, since 2000, in Valladolid, Spain, in different research groups dealing with multimedia processing, especially video, successively as a Research Engineer, Project Leader, and Division Manager. He is now working as a senior research consultant. His main interests are related to digital image and video processing, especially in the fields of image and video representation, content analysis for multimedia libraries, cataloguing and searching applications, multimedia and semantics, and metadata management and distribution. He has taken part in several European research projects, and is also involved in international standardisation initiatives, especially those of the MPEG committee.



**Robert-Paul M. Berretty** received a Ph.D. degree in computing sciences from Utrecht University (The Netherlands) in 2000. In 2001 he was a Postdoctoral Researcher at the University of North Carolina at Chapel Hill (USA). In 2002, he joined Royal Philips Electronics. There, he started as a Senior Scientist. Currently, he is cluster leader of the video processing cluster of the Digital Signal Processing group. He was coordinator of the OLGA project. His main interests include applied computational geometry, computer graphics and image processing, as well as object modelling in both still images and video streams. He is currently working on signal processing for 3D display applications.

