4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage (2003)
D. Arnold, A. Chalmers,F. Niccolucci (Editors)

Enhancing Virtual Reality Walkthroughs of Archaeological Sites

G. Papaioannou, A. Gaitatzes and D. Christopoulos.

Foundation of the Hellenic World

Abstract

This paper describes the methodological aspects of the application of various established and new graphics techniques in virtual reality applications, in order to visually enrich conventional walkthroughs and extend the common capabilities of virtual environment visualization platforms. The paper describes these techniques and goes to the extent of explaining various practical implementation issues. Examples and application case studies are provided to demonstrate the enhancements.

Keywords:

Virtual reality, lightmaps, image-based rendering, culling, video.

1. Introduction

Virtual reality is usually associated with large surround screen projection screen installations, stereo displays, head mounted displays and other high-tech hardware encountered in theme parks around the world. But virtual reality is in essence the art of illusion of simulating a fake environment where the spectator is immersed and cut-off from reality as effectively as possible, no matter the means to achieve this. Each application area has its own hardware and software platforms to support this alternative experience, depending on the role of the VR environment and the amount of interactivity the users are involved in. Naturally, VR is an attractive and intuitive means to visualise and explore the past and is thus exploited in education, tourism and research in the cultural heritage and archaeology domain.

Most archaeological applications of virtual reality are focused on exploring virtual reconstructions of sites, buildings, outdoor scenes and individual artefacts, with an educational motivation, although other archaeology related applications such as excavation planning and automated reconstruction tools do exist as well. Through the use of intriguing and visually appealing virtual worlds, people and especially young children, can effortlessly study and learn while being entertained. In our experience, Immersive environments have a significant impact on the memorization of information and the contextual linking of entities and meanings discovered in a virtual tour, rendering VR an effective educational tool.

Using cultural heritage content in a VR application implies that a minimum level of representation validity must be conveyed from the historical sources to the visual content.[1] Unfortunately, historical validity, although of crucial importance to an educational / edutainment / research scenario, is a limiting factor to the amount of visually stunning computer graphics and the nature of interactive story-telling. Visualising ancient worlds is a vivid example of this type of limitation due to the uncertainty about the exact appearance of buildings, clothing etc, due to lack of concrete and detailed evidence for many elements that appear in a virtual world. Imaginative creation in such an environment is not always allowed, because bringing the extremity and exaggeration of Hollywood-like productions to a cultural heritage application can and does spoil the value of the historical research behind the VR walkthrough.

Therefore, by avoiding building on too little information and distorting facts, archaeological walkthroughs can only highlight and visually enhance the virtual world by pursuing the convergence of photorealism and real-time rendering. It is true, that hardware technology in haptics, displays and graphics processing, reaches new levels of perfection and speed every day, but at the same time, the demand of the spectators is increasing with equal or more rapid rates. High detail representations encountered in pre-rendered graphics are too slow to render in real-time systems and the same holds true for the light interaction and visual effects. Instead, VR and real-time visualization requires carefully simplified models and fast and clever algorithms to brute force rendering of off-line systems. Textures replace geometric details, models are presented in multiple levels of detail, surroundings are displayed

only when they are visible and so on [2]. Evidently, programming and computer graphics theory techniques from the realm of consumer-level game industry have to be utilised even in large-scale visualisation installations to bridge the gap between the world of cinematic productions and real-time VR applications.

This paper summarises the attempt to visually enhance archaeological walkthroughs through the use of techniques already applied individually in the game industry and broadcast applications in an immersiveVR platform. At the same time, many optimisation techniques are presented as well, which are used to counterbalance the processing overhead of the visual enhancements. The reader should also bear in mind that an immersive VR application has the processing cost of a conventional 3D shoot-em up game with large polygonal environments multiplied by the number of display pipes involved in the surround screen setup, and all that doubled for stereo vision.

2. Enhancing Visual Credibility

When VR was first utilised for walkthroughs, the state of technology only allowed for single-pass-textured simplistic polygonal environments with simple triggered interaction. In the meantime, games and low-end computer systems have evolved, but these advances were not ideally followed on the more cumbersome VR platforms, mostly due to lack of competitiveness in the respective market.

Visual realism in computer games comes in many flavours but mostly relies on heavy texturing, special illumination effects, pre-computed illumination and detailed animation. A VR application, being essentially a game engine, can incorporate the technologies associated with these enhancements. Many commercially available visualisation libraries for high-end systems, like SGI's OpenGL Performer™ and OpenGL Shader™, already use standard techniques like multiresolution models and image-based rendering.

2.1. Pre-calculated illumination

Recently, new advances in graphics hardware have made possible the use of real-time shadows in computer games, although still at a severe computational cost. Fortunately, virtual archaeological worlds mostly comprise of inanimate environments and the interaction with the scenery is non-destructive. This way, dynamic illumination, including shadow generation is limited to moving characters and a few props (e.g. carts) that move in the vicinity of the virtual camera.

Most archaeological walkthroughs involve outdoor scenes with reconstructed buildings. Pre-calculated illumination in the form of illumination maps (lightmaps) [3] as double pass to standard material textures or standalone textures like baked or combined textures, has more visual impact and is cheaper than the shadow volumes shadow generation technique. Dynamic shadows are only needed for the few moving objects, when these shadows cannot be approximated by projective textures. Furthermore, outdoor scenes are lit by sunlight, which can be simulated by a combination of sharp-shadow, infinite point light source and a simple skylight direct illumination model. Most commercial modelling and rendering software supports baked texture generation and export mechanisms, although custom solutions can be adopted (see Figure 1 and the case studies section).

2.1.1. Memory and Texture Space Utilisation

The biggest issue with baked textures is the memory utilisation. Even in expensive graphics subsystems, texture memory is a valuable resource, which the

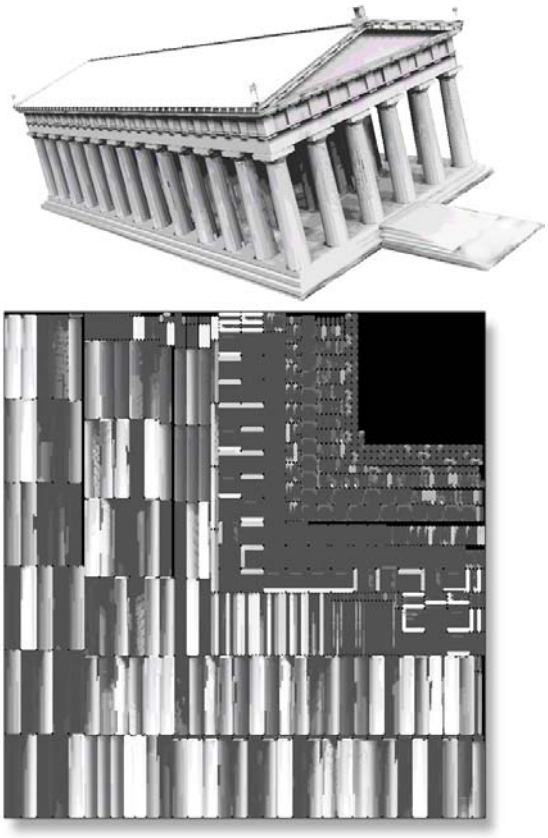

Figure 1: *Skylight illumination on a low detail model of the Temple of Zeus at Ancient Olympia, Greece. Below it, the lightmap used for texturing the column ring of the temple.*

programmer quickly runs out of. Texture baking is a good alternative to multi-pass rendering because it combines the material decals with the otherwise blended illumination second pass texturing. Unfortunately, the advantage of using tiled textures for repetitive patterns is lost, as the single pass textures that incorporate the illumination information need unique texture coordinates. This obviously leads to poor texture memory utilisation and a data transfer bottleneck when transferring textures to and from the display hardware memory. The apparent advantage of having a single pass instead of a double one is very frequently outbalanced by this texture swapping, especially in complex scenes.

Bad memory utilisation often occurs from poor texture packing. Lightmaps are built for individual polygon clusters, after segmenting the initial geometry into patches of adjacent triangles, using the triangles' normal vector direction as a splitting criterion [4]. The lightmaps are then combined into a single texture and the mesh vertices are assigned unique texture coordinates. This process involves "packing" or stacking the lightmaps into a single image using an image space subdivision method [5]. An initial estimate of the size of the individual lightmaps can be determined using the ratio of the area of each lightmapped polygon cluster (or the axis-projected area, if axis-oriented splitting is used) against the total surface area. Unfortunately, the texture packing procedure may leave unused space between the lightmaps in the final image, resulting in a failure to include all maps into a single texture. One solution is to use a second texture map and store the remaining textures there, but as texture memory is a limiting factor, it would be preferable if all lightmaps could be packed within a single, user-constrained texture.

We achieve this by incorporating in our custom lightmap renderer an iterative lightmap resizing mechanism, which operates before actually rendering into the lightmaps: Relative lightmap sizes are determined using the projected polygon area of the polygon clusters as mentioned above. Then the lightmaps are repeatedly scaled and packed into a singe texture until they fit as tightly as possible without excluding any lightmap (Figure 1). This way, illumination maps get the maximum crispness possible for a given texture size and no valuable texture space is wasted in blank image space.

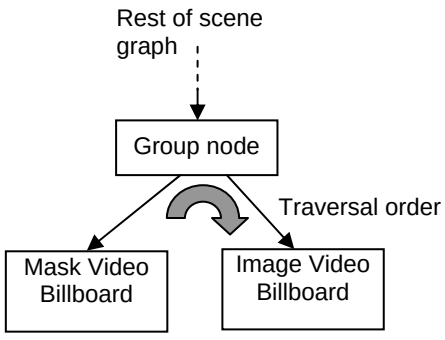

Figure 2: *Video Overlay on billboard geometry in a scene graph.*

2.2. Video Source Overlays

Merging 3D worlds and real action is a common practice in cinematography. Chroma-keyed video sequences of actors and props filmed in front of a blue screen are easily overlaid and mixed with pre-rendered animation sequences and imaginary scenes. But what if this experience should be transferred to the VR set? Real actors may participate as virtual guides into the reconstructed world or be a part of the background activity of a marketplace and give an impression of familiarity, natural movement and look to the polygonal scenery [6].

The biggest problem is that most video formats do not provide an alpha channel (transparency) and, worse, most media libraries do not support this extra information, not to mention cross-platform portability issues. Therefore, it is difficult to infuse a blending mechanism in the video itself. Porting the video post-process notion of chroma-keying to a real-time rendering system is also a poor solution. An acceptable quality video source, especially for close-up inspection (e.g. a virtual guide) cannot be of a resolution less than 256X512 pixels wide for a figure or object covering 1/4-1/3 of the viable screen area. Filtering this image to derive an alpha value from the RGB hue at a frame rate of 15-30 fps is too demanding a process to have on a VR system already struggling to render all the textured geometry and handle the game engine events.

In our implementation, we have opted for an OpenGL® blending mode that uses double pass rendering to simulate masking [7] (Figure 2). The masking operation is as follows: An inverted mask is generated from the original source, which is white where the image should appear transparent, and black in the solid areas. Additionally, the background colour of the original image (e.g. blue for a blue screen) is blackened. The mask is rendered first with a disabled depth test and a blending function of (GL_DST_COLOR, GL_ZERO). Then, the normal image is rendered with a blending function of (GL_ONE, GL_ONE) on top of the mask.

This technique was originally applicable to low-level immediate OpenGL® rendering, but the idea was ported to a scene-graph-based environment, based on OpenGL Performer™. The original video is post processed to fit a power-of-two size, as the video source is used as a texture. Using chroma-selection, an inverted

Figure 3: *A Video source (girl) is superimposed as a billboard in the scene. The partial silhouette haloing is a result of the chroma-keyed transparent cloth and hair.*

masking video is produced and saved in a separate video file. In each frame draw call, a time-synchronised frame is loaded from the buffer of both channels (mask and image) and placed on an identical piece of geometry. In the case of our VR engine, a billboard was adopted as the underlying geometry (Figure 3), as the original use scenario demanded an image of a real actor always moving in front of the user, in a guided tour.

2.3. Image-Based Modelling

High geometric detail is a key factor for the generation of a convincing virtual world. For parts of the scene that are far away from the observer or objects that the user does not interact with, it is a common practice to replace the geometric complexity with textures [8]. An important psychological aspect of the VR walkthrough is that by constantly moving through the virtual world, the representation accuracy is mostly felt rather than meticulously observed. Therefore, if heavy, life-like texturing replaces detailed architectural models in an archaeological site, the image is going to "fit in" the senses and the impact is going to be greater than overloading the system with modelled details that no one is going to get too close to notice.

Extending this idea, image-based modelling can be used for the generation of simplified geometry from photographs of real archaeological sites or objects. The big advantage is that one gets at the same time a low

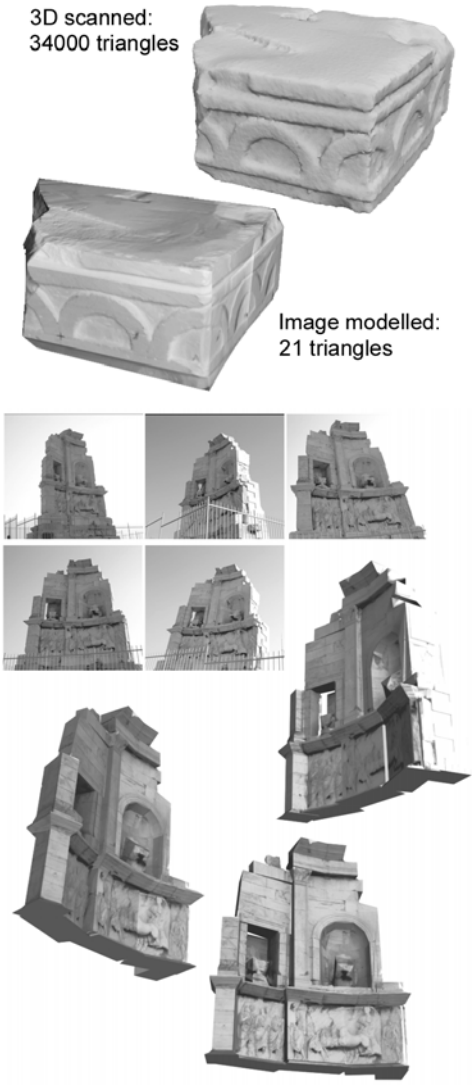

Figure 4: *Image-based modeling. Above: Visual detail is acquired with conventional photography and imprinted as textures on simple geometry. Below: The monument of Philopappou in Athens. Textures based on real pictures of the site cover a rough approximation of the monument's geometry.*

polygon representation of the scenery and properly registered appearance and illumination maps, directly from the recorded imagery (Figure 4).

Image- or photogrammetry-based modelling works well with both distant outdoor entities, like tree lines and whole buildings, and closely inspected crude objects like slabs and columns. The main disadvantage is the view dependency of the texture maps, which implies that in

order to avoid distortion or exposing unregistered and untextured parts of the scene, one has to restrict the navigation activity close to certain paths. Evidently, image-based modelled objects have as low a polygon count as the modeller wishes but on the other hand, they suffer from the same restriction that applies to the illumination maps: The textures derived from the image capture and warping procedure cannot be tiled, thus increasing the memory utilisation of the graphics subsystems.

The procedure of extracting the geometry from a sequence of images itself is a fairly known but tedious semi-automatic task that involves defining and calibrating virtual cameras in 3D space that correspond to the photographs taken and the detection of common reference points in the original photographs [9]. Using computer vision methods, these points are back-projected in 3D space and can be used as building reference points to model a crude approximation of the observed geometry. By projecting, merging and clipping the original photographs on the polygons generated by hand, one can get the textures required to display the original objects in a convincing manner.

2.4. Detail Textures

Another technique for adding visual detail in a virtual scene and more precisely to the terrain and tiled materials, is the use of detail textures. They are special textures used to give some sense of detail when viewed from very close. When the main texture starts to get blurry from being viewed too close to it, the detail texture gradually blends in with the coarse layer. A detail texture is a very small, fine pattern which is faded in as you approach a surface, for example wood grain, or imperfections in stone. Detail textures modulate the surface colour they are applied to, this way scaling the surface's brightness up or down. Colour components in the range 0-255 with RGB brightness values from 0-127 darken the surface; 128 has no effect; and 129-255 brighten the surface. Therefore, when drawing detail textures, it is important that you design them with their brightness in the proper range so they affect surfaces naturally. If a detail texture's average brightness deviates too far from 128, then surfaces will appear to brighten or darken as you approach them.

A second rendering pass of the *detailed* geometry is usually required, unless multi-texturing is natively supported and this affects frame-rate, so it is a judgement call when to use them and when not to. For outdoor terrain however, where, because of the walkthrough navigation mode, the viewpoint is always close to the textured ground surface, detail textures substantially improve the crispness and natural look of the image; the base texture alone, no matter how large, cannot stand a very close examination. In practice, a small 64X64 or 128X128 detail texture with Gaussian monochromatic (greyscale) noise samples centred at 128 RGB intensity has a very good visual impact.

3. Speed Optimisations

3.1. Image-Based Rendered Characters

The rendering of outdoor and indoor environments requires the synthesis of what are often considered two separate problems: the real-time visualization of large scale static environments, and the visualization of characters and probs.

In archaeological and walkthrough applications most of the bandwidth and rendering capacity is used for the outdoor and indoor environments and buildings, which are usually the points of interest. In order to enrich these environments characters and other props might be used. To render many animated characters and props as complex polygonal models will result in non interactive frame rates, therefore a balance has to be made between the amount of models used and their complexity. Unfortunately using exclusively low polygon models comes into contradiction with the rest of the environment, which might be of better quality and is therefore bound to break the suspension of disbelief.

In order to minimize geometric complexity and achieve a balance between amount and quality of models, image based rendering and impostors are often used for specific models. Especially human representations have been successfully modelled using image-based approaches [10]. The principle of image based rendering techniques is to replace parts of the polygonal content in the scene with images. These images can be computed dynamically, as is the case with relief texture mapping [11], or a priori and are then placed as impostors into the environment using simple geometry like planes, which always face the viewer. Usually the images for the impostor are taken from different positions around the original object and are selected at run-time depending on the viewpoint position and the frame of animation.

In our recent application "Feidias' Workshop" we implemented IBR techniques for visualization of animated characters and discovered their usability and impact in VR and educational applications. Each character was replaced by an impostor, whose texture was changed depending on the viewpoint position and the frame of animation (Figure 5). Thus there was information for all the possible views of the impostor. The geometry used for the impostor was a quadrilateral facing always the user (billboard). Pre-rendered view-dependent textures were preferred instead of dynamic techniques like relief textures due to the serious computational overhead imposed on the processors, especially in the case of multiple IBR nodes.

Figure 5: *Image-based rendered characters from one of our recent applications.*

3.1.1. IBR advantages

Using IBR impostors has various positive impacts on the application if implemented properly. Since the rendering of the images is usually done offline the rendering speed in real-time applications is significantly faster in comparison to using polygonal models.

The original models from which the images are taken can be detailed, high-polygon-count models. These models, although being too expensive for real-time use, are perfect candidates for IBR techniques, since the geometric complexity does not deter the offline renderer and provides convincing visual details for the illumination interplay. This possibility of high-detail has been very beneficial in educational and archaeological applications. The human eye is accustomed to detect details on depicted human figures and therefore, complex pre-rendered imagery increases their photo-realism and natural appearance.

Modelling human characters for real-time use is a difficult and daunting job, which only skilled modellers and artists can perform efficiently. A low polygon character aims at being efficient in structure, polygon count and texture usage, while at the same time avoids sacrificing too much quality. This trade-off involves careful and skilled modelling. Models needed for IBR do not have these restrictions since they are rendered offline. Often commercial packages can be used to construct an initial high-polygon human model. Alternatively, the source for the images used on the impostor need not necessarily be modelled, but instead, they can also come from real photographical images or video captures.

Using impostors has proven to be ideal for characters that perform an isolated action in the background and which the user does not interact with or gets too close to. View-dependent impostors exploit frame-to-frame coherency efficiently and require less viewing angle instances if they are always far enough from the viewer. This way, there are slow and constrained changes to the visible sides of the depicted subject, thus requiring a sparse sampling of the viewing direction space.

Animation for characters rendered using IBR techniques can be realized faster and incorporated easier into an existing engine or framework than other full 3D animation techniques like morphing or kinematics. It is highly applicable for filling the environment with activity making it interesting and avoiding the typical VR walkthrough in empty environments with buildings and static trees.

3.1.2. IBR pitfalls and disadvantages

Although the use of IBR impostors may seem easy at first, it also has certain pitfalls and difficulties, which also apply to the video textures. The most annoying artifact when using impostors is the popping that occurs in the transition from on image (view direction) sample to another. Because the image samples were taken from certain positions there is no information about the depicted geometry for viewing positions in between them. This results in sudden and abrupt changes.

When getting too close to the impostor, pixelisation often occurs. The bigger the texture used the better the image quality and the more the texture usage on the graphics system. Using too small textures makes the image look blocky and using too big textures drains the bandwidth with texture downloads.

Because there is no information for the Z-buffer when rendering the impostor, rendering artifacts where the impostor plane intersects 3D geometry can occur, so careful scene design is important, in order to leave enough space around the impostor to allow it to freely rotate without bumping on the surrounding objects.

3.1.3. Overcoming IBR disadvantages

To mitigate the problem of popping between the image samples, a single-pass multitexturing algorithm is applied which blends the closest impostor view instances together.

In order to minimize texture usage and texture bandwidth, tests have to be performed about what texture sizes to use, so as to maximize the useful area that an image object occupies. The texture size depends on the object itself and the usage of the impostor in the scene. Impostors further away can have smaller texture sizes.

For easy production and preview of the impostors a special tool was developed. This IBR model pre-processor takes into account the desired texture size, the 3D model extends and the number of positions and elevations (rings) around the model from where the images would be captured. The software enables the user to interactively position and scale the model inside a window that has the size of the specified texture and also preview the model in various positions from where a capture will be made. This way, it is easy to place the model so that it occupies the maximum area of the image window without intersecting the image boundaries.

When rendering the images of the impostors, multi-sample antialiasing is used in order to reduce the pixelisation in the image texture and enable the usage of smaller, smoothed-out textures.

The view-dependent image sample density can be optimised with respect to the viewable range and number of samples for each impostor separately. The character or depicted object may be sampled from certain views from which it is known to be observed more frequently. For instance, if the character is to be viewed only upfront and rarely from above, viewpoint elevation ranges between -20 and 20 degrees should be more densely sampled than the rest, which, in some cases (see section 5) may be not sampled at all. This technique imposes some complexity on the rendering and transition between samples since each elevation can have different amount of views.

3.2. Culling

In order to represent a VR Cultural Heritage scene in great detail as to show its attributes, a great deal of polygons is required. Simple rendering algorithms though cannot cope with the plethora of such polygons for a real time walkthrough of the archaeological site. Techniques are required to eliminate polygons even-though they would normally be rendered since they exist in the viewing field of the user. Different intelligent culling techniques exist that remove polygons from the rendering pipeline.

3.2.1. Shadow Culling

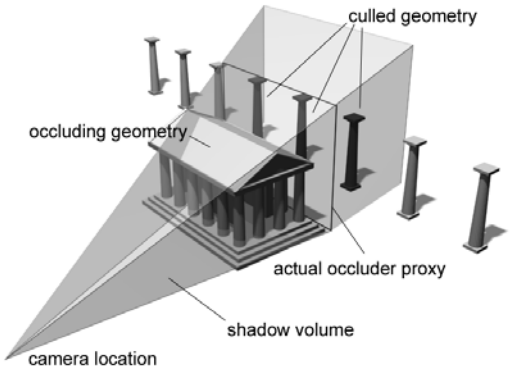

Figure 6: *Shadow Culling technique for eliminating non- visible polygons in the viewing frustum.*

Hudson et al. [12] proposed a geometry culling method where the main idea is to select a small set of large occluders and discard the objects behind the occluders, i.e., objects that are *shadowed* by an occluder with respect to a certain viewpoint. These occluders may not be regular visible scene geometry, but also invisible mattes purposefully inserted in the virtual world for culling other geometry. Culling is performed by calculating the shadow volume pyramid that stems from the viewer position, passes from the occluder polygon boundaries and extends to infinity and determining which sets of objects fall completely inside this volume (Figure 6). These objects are not drawn at all for the particular frame. In average, the extra culling calculations performed by the rendering engine take far less time than rendering the hidden geometry and about 20% is gained in rendering speed.

3.2.2. Contribution Culling

Another culling technique is contribution culling [13], in which objects whose visual contribution is smaller than a user-supplied threshold are discarded from consideration even though they may be visible. The factors defining contribution are generally object size and range, field-of-view, display window size, and monitor resolution.

Generally, contribution culling can be thought of as part of the level-of-detail simplification process. The modeller - scene designer determines manually the distance from the viewer that certain objects could vanish completely from sight, according to the object's size and set an empty LOD target at this range. Trees and other scenery parts of trivial contextual importance can be treated with contribution culling. This technique is very valuable when exploring outdoor archaeological sites, because there are many locations of interest that draw the eye and help vanishing geometry to pass unnoticed.

4. Implementation

Employing VR components that are user-friendly and easy to use creates an important base for the software developed to use this hardware. The software provides a layer of mediation between the hardware and the final programmer and user; VR applications are usually developed using object-oriented languages on top of graphics libraries such as SGI's OpenGL Performer™ and OpenGL®.

All enhancements and extensions discussed above are built on top of the VR framework initially developed by Pape et al [14] and adapted for the cultural heritage applications of the Foundation of the Hellenic World [1]. The engine development was done in C++ and was based on OpenGL Performer™ and OpenGL® for the graphics, on the CAVElib™ library for transparent access to handling virtual reality hardware and components and a custom sound library for playing positional audio (Figure 7). These libraries provide the abstraction layer necessary for the execution of the runtime environment on all VR hardware platforms used by FHW.

The system is divided into two major components: the virtual world authoring interface, consisting of a scripting language parser and conversion tools, and the core VR engine which runs the compiled scene-graph of the VR world and its environment and behaviours in real time.

The new engine extends the capabilities, such as particle systems, morphing and dynamic texturing, of the original one with lightmaps, real-time video overlays, view dependent textures and object-to-object collision detection. New productions make use of the aforementioned culling techniques and introduce image modelled elements and view-dependent sprites in the visual content.

In truth, not all enhancements affect the core engine. Lightmaps are anyway incorporated as a multi-texturing stage in the rendering process, regardless of the nature of the depicted information. Commercial software often provide the illumination maps along with the geometry, which can be readily displayed by the engine. Sometimes though, custom illumination (e.g. skylight or cartoon models) must be deployed, or older – or third party - models must be illuminated without affecting the original geometry. For these situations, a custom lightmap generation tool, has been developed. The tool is built on top an extensible shader and illumination architecture to accommodate future needs. It uses an iterative, adaptive lightmap-space ray-tracing algorithm for tracing incident irradiance from the sampled light sources and the environment. Currently, direct illumination with a wide range of light sources, including a skylight model, is supported, while future versions will include radiosity, dust and corrosion simulations.

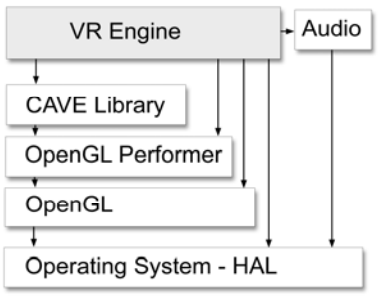

Figure 7: *VR engine layers*

5. Case Study: The Workshop of Feidias

Feidias' Workshop (Figure 8) is an interactive virtual experience in the laboratory of the sculptor Feidias. It takes place in ancient Olympia, amongst an accurate reconstruction of the famous statue of Zeus and the sculptor's tools, materials, and moulds used to construct it. Visitors become the sculptor's helpers and actively participate in the creation of the huge statue, by using virtual tools to apply the necessary material onto the statue and add the finishing touches.

Since this application reconstructs the interior of a workshop the technique of lightmapping, which is very

Figure 8: *Image from Feidias' Workshop where most of the techniques described are visible.*

well suited for indoor environments, was used for shadows and lighting simulation on the static environment. The statue of Zeus, the tools and the interaction with them had to be exact and detailed. This resulted in high polygon counts and high texture usage. Although all the necessary models and tools were incorporated in the scene, the absence of human characters was evident.

We chose to incorporate characters depicting workers and Feidias, the sculptor itself, using IBR in the form of animated impostors. This technique was chosen because of its low polygon count since the detail of the models in the scene was high. Another reason was because it would bind nicely with the existing lightmapped shadows. Lightmaps use static light sources. By using the same light sources for the impostor images, the detail of shading in the images would match completely with the shading in the workshop producing consistent and realistic results.

The models and some animations were created using Curious Labs POSER© and Discreet 3Dstudio Max© in a fraction of the time compared to morph target 3D models we created for full 3D character animations in previous projects. The texture sizes used was 128x256 and 256x256 depending on the pose of the character. Despite this being a very low texture size, because of the blending and antialiasing techniques used to create the final images, the impostors looked quite satisfying even from up close.

By careful positioning the characters the possibilities of artifacts when viewed from various angles, which may be caused by intersection of the impostor plane with the 3D models, were avoided. Furthermore we noticed that usually the characters were seen from upfront. Therefore the views perpendicular to the character (elevations between -20 and 20 degrees) were optimised using more sample images. In some cases it was sufficient to have sample images only for these views, thus providing no sample images for other elevations. Due to the fact that each impostor had a very sparse set of images associated with it, we opted for a closest neighbour selection criterion of the view-dependent textures rather than a blending mechanism. Interpolation between inconsistent views tended to produce blurry results and, as the characters were mostly viewed from arbitrary angles, they always had an insubstantial appearance.

The result was a reconstruction of a populated and active workshop in which the visitors would also observe and learn about the habits and appearance of workers during that time. The use of characters even as impostors enriched the environment tremendously making it an interesting experience for all ages and types of visitors.

References

1. A. G. Gaitatzes, D. Christopoulos and M. Roussou, "Reviving the past: Cultural Heritage meets Virtual Reality", *VAST2001: Virtual Reality, Archaeology, and Cultural Heritage*, November 2001.

2. T. Akeninne-Möller, E. Haines, *Real-Time Rendering*, A K Peters, 2002.

3. J. Arvo and D. Kirk, "Fast Ray Tracing by Ray Classification", *ACM Computer Graphics* (SIGGRAPH '87 Proceedings), 21(4):55-64, 1987.

4. G. Papaioannou, E. A. Karabassi and T. Theoharis, "Segmentation and Surface Characterization of Arbitrary 3D Meshes for Object Reconstruction and Recognition", *Proceedings of the IEEE International Conference on Pattern Recognition '2000*, pp. 734-737, 2000.

5. S. Zhukov, A. Iones and G. Kronin, "Using Light Maps to Create Realistic Lighting in Real-Time Applications", *Proceedings of WSCG '98*, pp. 45-55, 1998.

6. A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, "Video Textures", *Proceedings of SIGGRAPH 2000*, ACM, pp. 489-498, 2000.

7. The Neon-Helium Productions Tutorials: http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=20

8. P. Sander, X. Gu, S. Gortler, H. Hoppe and J. Snyder, "Silhouette Clipping", *Proceedings of SIGGRAPH 2000*, ACM, pp. 327-334, 2000.

9. O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, 1993.

10. F. Tecchia, C. Loscos and Y. Chrysanthou, "Image-Based Crowd Rendering", *IEEE Computer Graphics & Applications*, 22(2):36-43, 2002.

11. M. M. Oliveira, G. Bishop, D. McAllister, "Relief Texture Mapping", *Proceedings of SIGGRAPH 2000*, ACM, pp. 359-368, 2000.

12. T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff and H. Zhang, "Accelerated Occlusion Culling using Shadow Frusta", *Thirteenth ACM Symposium on Computational Geometry*, June 1997.

13. DirectModel 1.0 Specification. Hewlett-Packard, October 1997.

14. D. Pape, T. Imai, J. Anstey, M. Roussou and T. DeFanti, "XP: An Authoring System for Immersive Art Exhibitions", *Proceedings of VSMM '98, November 1998.*