The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST (2005) Short Presentations
M. Mudge, N. Ryan, R. Scopigno (Editors)

High Fidelity Walkthroughs in Archaeology Sites

Luís Paulo Santos[†], Vitor Coelho, Paulo Bernardes[2] and Alberto Proença

Departamento de Informática, Universidade do Minho, Portugal
[2] Unidade de Arqueologia, Universidade do Minho

Abstract
Fast and affordable computing systems currently support walkthroughs into virtual reconstructed sites, with fast frame rate generation of synthetic images. However, archaeologists still complain about the lack of realism in these interactive tours, mainly due to the false ambient illumination. Accurate visualizations require physically based global illumination models to render the scenes, which are computationally too demanding.
Faster systems and novel rendering techniques are required: current clusters provide a feasible and affordable path towards these goals, and we developed a framework to support smooth virtual walkthroughs, using progressive rendering to converge to high fidelity images whenever computing power surplus is available.
This framework exploits spatial and temporal coherence among successive frames, serving multiple clients that share and interact with the same virtual model, while maintaining each its own view of the model. It is based on a three-tier architecture: the outer layer embodies light-weight visualization clients, which perform all the user interactions and display the final images using the available graphics hardware; the inner layer is a parallel version of a physically based ray tracer running on a cluster of off-the-shelf PCs; in the middle layer lies the shading management agent (SMA), which monitors the clients' states, supplies each with properly shaded 3D points, maintains a cache of previously rendered geometry and requests relevant shading samples to the parallel renderer, whenever required.
A prototype of a high fidelity walkthrough in the archaeologic virtual model of the roman town of Bracara Augusta was developed, and the current evaluation tests aimed to measure the performance improvements due to the use of SMA caches and associated parallel rendering capabilities. Preliminary results show that interactive frame rates are sustainable and the system is highly responsive.

Categories and Subject Descriptors (according to ACM CCS):
I.3.7 [Computer Graphics]: Ray Tracing I.3.2 [Computer Graphics]: Distributed Network Graphics
J.2 [Computer Applications]: Archaeology

1. Introduction

The ability to navigate through a model of an archaeological site allows researchers to visualize virtual reconstructions of long disappeared sites of interest. This ability becomes even more empowering for the researcher if the navigation is generated with high fidelity rendered images (where light interreflections among objects are included on the illumination model), is interactive (geometry and materials' properties can be changed by the user), and is collaborative (many users share the same model and can visualize and comment each others changes to the model). This communication mainly addresses the quality and frame rate generation of rendered images required to an walkthrough in an ancient roman town, Bracara Augusta, while the overall framework also cares for interactive and collaborative work.

Bracara Augusta, currently Braga, Portugal, was one of the three major towns founded by Emperor Augustus in the Iberian Peninsula, at the end of the Cantabrian wars (around 16 BC). The archaeological rescue project of Bracara Augusta started in 1976: some relevant buildings and infrastructures, regarding urban development and architecture, have been found, studied and interpreted. Based on these archae-

[†] partially supported by portuguese FCT project POSI/CHS/42041/2001

© The Eurographics Association 2005.

ological data and on their interpretation, a first challenge to virtually reconstruct Bracara Augusta resulted on a virtual model, which shows the urban development of the city with some accurate reconstructions of the Alto da Cividade thermae (the only public health-resort totally excavated in Braga), the insulae of the Carvalheiras (a wealthy private house), the defensive wall and the orthogonal streets. The virtual model of Bracara Augusta was created in 2000, using some available commercial tools. It has a complexity of about 1,140,000 faces; to increase the realism of the model some parts were textured and a global (although not realistic) illumination model was created. One of the main goals of Bracara Augusta's virtual model was to make an educational movie to promote archaeological research. Each frame of the 20 minutes long movie was rendered in about 2.5 hours on a SGI Octane, single processor [MB00].

Fast and affordable computing systems currently support frame rate generation of synthetic images and the possibility to navigate into these sites. However, archaeologists complain about the odd feeling they experience in these interactive tours: lack of realism, mainly due to the false ambient illumination, which does not take into account light interreflections among all objects. To take advantage of the walkthrough facilities, archaeologists require correct illumination to study the architectural building options and the urban development of the town. This includes to observe how the behaviour of the sun influences the size and shape of windows and the allocation of the inner space; to understand the quantity of day-light in every house compartment, enhancing the perception of realism in the inner space of the building; to infer if a room needs artificial lighting (oil lamps) and simulate these artificial lighting conditions. To provide accurate visualizations of the underlying virtual world, physically based global illumination models must be used to render the scenes. Global illumination algorithms, however, are computationally too demanding to allow interactive navigation on large virtual models using current PCs.

Brute force rendering of each frame on a virtual walkthrough is impractical and ingenious techniques are required to support a smooth navigation while presenting tolerable degradation of the image quality. To support smooth virtual walkthroughs a technique based on spatial and temporal coherence among successive frames is exploited, while resorting to progressive rendering to converge to high fidelity images whenever there is available computing power. It is structured as a three-tier architecture: a parallel renderer, a Shading Management Agent (SMA), which caches previously computed shading values, and visualization clients.

Future PC generations will be able to deliver such services, since their CPU architecture is moving into multi-core CPUs per chip, increasing execution parallelism, required to boost the expected performance for real-time generation of high quality images. Current R&D on the deployment of parallel and scalar applications on computing clusters, is a feasible and affordable path towards these future generation systems. A prototype of a navigation system to walk through the virtual model of Bracara Augusta was developed, which implements some of these techniques and takes advantage of a computing cluster.

The next section presents the framework used to support smooth high fidelity virtual walkthroughs. The following section contains an overview of the three-tier architecture. To evaluate the frame rate generation of high quality rendered images (without progressive refinement), section 4 describes the experimental methodology and setup, while section 5 critically analyzes the measured times. The conclusion section closes the paper.

2. Algorithm Overview

To support high frame rates both spatial and temporal coherence are exploited. This approach relies on image space subsampling and object space caching of shading values; progressive refinement is proposed as the mean to converge to high quality images, whenever computing power surplus is available. Two basic functionalities of a navigation system are identified and decoupled: image visualization and rendering run asynchronously and at their own pace. An additional process is identified and inserted between the renderer and the visualizer: a "Shading Management Agent" (SMA) computes visibility and decides which, and whether, shading samples are requested to the renderer or retrieved from a local cache of previously evaluated shading information (see figure 1). This work is based on the ideas presented on [TPWG02].

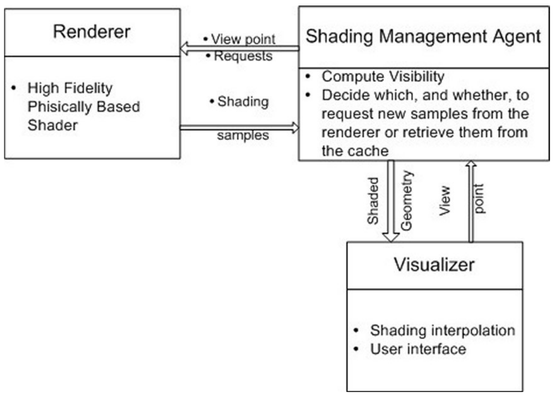

Figure 1: System structure.

Spatial coherence is exploited by subsampling the image space, requesting accurate shading samples only at visible triangles vertices and interpolating among samples when displaying the image [TPWG02, WDP99, WDG02]; interpolation is performed by the visualization client, using the graphics hardware. The SMA determines which triangles are visible for a given view point and either requests vertices' shading values from the renderer or retrieves these values from the local cache, if available. By supplying the visualizer with only the visible portion of the original geometry, the traffic volume between the SMA and the visualizer is reduced and the workload imposed upon the visualizer hardware is kept to a minimum. Furthermore, since the visualizer is supplied with the original geometry it is always geometrically accurately reproduced, even if the view point changes. Shading values resulting from specular interactions, however, become incorrect with view point changes, and must be recomputed (refreshed) frequently. Image reconstruction from a sparse set of samples is inaccurate on those regions where high spatial frequencies occur, such as at shadows boundaries, specular materials (e.g., glass, mirrors) and on texture mapped polygons. This can be overcome by progressively requesting more samples to the renderer. The geometry is represented on the SMA using a hierarchical data structure, which allows original triangles to be arbitrarily subdivided. Whenever there is available computing power surplus (e.g., when the user stops on a given view point for some time), the SMA selects some triangles for subdivision according to a given set of criteria [BDT99, BWG03, FP04, WDG02] and requests shading values from the renderer. These subdivisions and respective shading information are stored on the hierarchical data structure and sent to the visualizer. The image will thus converge to a high quality image, given enough computing time; no interpolation will take place when each fine grain triangle maps onto a single pixel. Shading values refreshing and progressive refinement are not implemented on the prototype version under evaluation.

Temporal coherence is exploited by caching previously computed shading samples and reusing them whenever the geometry becomes visible again. Caching occurs on object space to avoid reprojection artifacts. The SMA is responsible for maintaining this local cache. Illumination artifacts occur, however, due to specular phenomenons. The cached samples must therefore be periodically refreshed. Temporal coherence is also exploited to reduce the communication traffic between the SMA and the visualizer by resorting to an incremental communication protocol. On an ordinary walkthrough most triangles visible on a given frame will also be visible on the consecutive frame; instead of sending all the geometry for each frame, the SMA indicates to the visualizer which new triangles must be added to the visible set and which ones are no longer visible. This results on three sets of data to be sent: the list of triangles to add that were found on the SMA cache, the list of triangles that are no longer visible and can be deleted from the visualizer data structure to reduce memory requirements and finally the list of newly visible triangles whose shading information had to be requested to the renderer. This ordering results in the fastest updates on the image displayed by the visualizer.

3. System Overview

The proposed system is based on a three-tier architecture: the renderer, the SMA and the visualizer. These functional entities can be mapped onto different sets of machines, favouring remote visualization and parallel computing.

A point based renderer, such as path tracing or ray tracing, must be used to accept requests arbitrarily distributed across the image plane or the object space. A modified version of Radiance [War94, WS98] is used, which accepts input over POSIX sockets and launches several processes across a cluster of workstations to increase rendering throughput. Radiance is a high quality physically based ray tracer extended with the irradiance cache to account for diffuse interreflections [WH92].

The SMA is a multithreaded program that connects to the renderer and the visualizer. The current evaluation version supports a single visualization client, while future versions will be able to accept connections from multiple clients, which share the same geometric model and cached samples, thus reducing memory and workload requirements. Special care must be taken, however, with shared shading values resulting from different view points, since the specular components depend on these. The system is conceived as a "Rendering Service Provider", in the sense that visualization clients can connect to the SMA from remote locations and can run on light, affordable machines. The SMA and the rendering processes can run on powerful machines, or even on a cluster of workstations, hosted on the service provider facilities and eventually shared by multiple clients.

The visualization client is a light-weight program with modest memory, computing and communication requirements. Care has been taken to assure that only the visible geometry is stored on the client's memory, thus reducing memory demands. Ordinary graphics hardware is required to project and interpolate over the geometry. An incremental communication protocol is used to minimize bandwidth requirements. With current developments on mobile devices technology it is conceivable that in the near future the client can run on portable devices, such as PDAs or mobile phones.

4. Experimental Setup

One of most relevant feature introduced in this prototype - the SMA cache - can introduce impressive speedups in the generation of consecutive rendered images, provided the new frame contains minor differences from a previously visited place. These are the main experimental results worth discussing in this paper, based on an walk through Bracara Augusta, with both large and short displacements of view points.

The selected virtual trip goes through 9 frames (see figure 2), with the following relevant features:

- this is not a typical trip, since large displacements impairs

a smooth generation of rendered images; it aims to illustrate some worst case conditions;
- from frames 3 to 4, 4 to 5 and 5 to 6, a large temporal coherence is expected due to the short view point displacement, while a reduced temporal coherence is expected between the other pair of frames;
- the view point in frames 6, 7 and 8 is very similar to frames 3, 2 and 1, respectively, thus an high SMA cache hit ratio is expected;
- frame 9 is a complete repositioning of the view point;
- a second walk through the same frames is performed, to stress improvements in response times, when all shading values can be retrieved from the SMA cache.

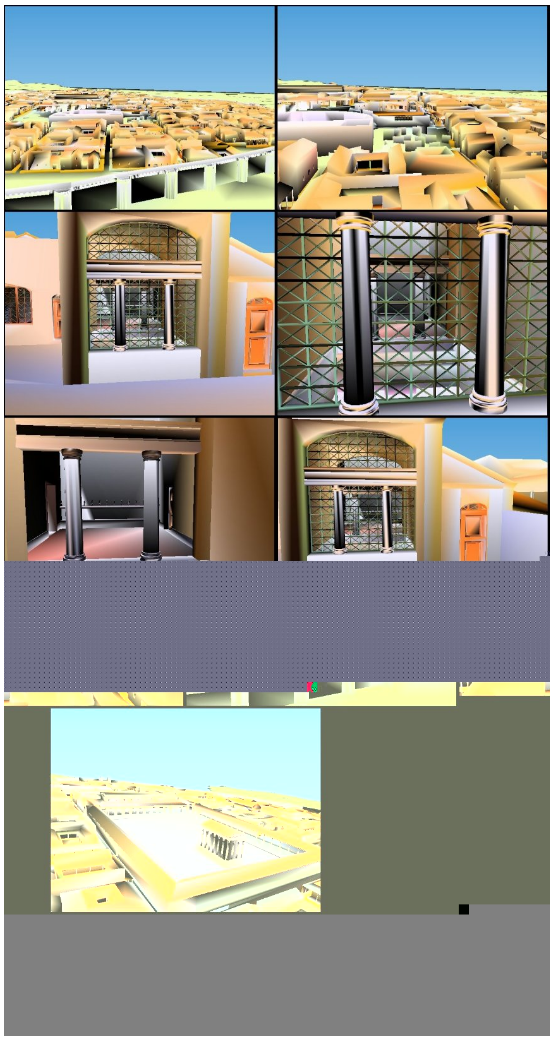
Figure 2: Walkthrough images: left to right, top to bottom.

The working environment used several computing systems, interconnected through 100 Mbit/s Ethernet:

- the visualization client was on a PC with an AMD Athlon CPU and a NVidia GeForce 5200 GPU;
- the SMA and one rendering process was on a dual-Xeon, 3.2GHz, 2GB RAM and a NVidia 6800GT GPU;
- the remaining parallel renderers was executed on a set of 3 AMD Athlon-based systems, 1.4GHz, 1GB RAM (to be soon replaced by 4 dual-Xeon, 3.2 GHz, 2GB RAM computing nodes)

The Bracara Augusta model has 1,140,223 triangles; each rendering process requires 1.1GB RAM to load the full model, using Radiance internal data structure. Radiance memory requirements exceed the physical memory available on the rendering nodes, requiring the use of virtual memory; this strongly penalyzes the speedup obtained with parallel rendering.

Table 1 presents the number of triangles found in the visualizer, in the SMA cache, and requested to the renderer the first time this path is traversed.

Frame	Total	Viz	Cache	Renderer
1	8455	0	0	8455
2	3731	2209	312	1210
3	3475	223	47	3201
4	3246	1816	405	1025
5	1662	521	431	710
6	2879	245	2425	209
7	3732	162	3178	392
8	8455	2223	5312	920
9	4579	322	104	4153

Table 1: Number of triangles on each frame: total, on the visualizer, on the SMA cache, requested to the renderer.

5. Results' Analysis

Table 2 presents the measured timings with a single rendering process on the same machine as the SMA. Times are presented in msec for the first and last data packets received with triangles retrieved from the SMA cache, T^{1st}_{cache} and T^{last}_{cache}, and for the first and last data packets received with triangles just shaded by the renderer, T^{1st}_{rend} and T^{last}_{rend}. The achievable frame rate is also presented. The first nine rows correspond to the first traverse of the designated path, while the last nine rows, F^2_1 to F^2_9, correspond to the second pass, where no samples are requested to the renderer. These results show that the physically based renderer is the main bottleneck, since T^{last}_{rend} increases, in average, with the number of requested samples. The relationship between these two is not strict, in the sense that more requests do not necessarily mean a longer rendering time; this is due to exploitation of coherence in Radiance's irradiance cache [WH92]. In fact, frame 2 takes longer to render that frame 3, although it requires less shading samples: a sparser distribution of the requests over object space results in more extended calculations of irradiance, rather than interpolating using previously cached irradiance values. The overall trend, however, is for rendering times to increase with the number of requests. Parallelism at the renderer level is thus an obvious solution.

Comparing the results between the first and the second passes, it is clear that, once all shading samples are in the SMA cache, the frame rate increases significantly, ranging from 3.4 to 20 frames per second.

Frame	T^{1st}_{cache}	T^{last}_{cache}	T^{1st}_{rend}	T^{last}_{rend}	fps
F_1	—	—	426	39089	0.025
F_2	58	58	201	3101	0.322
F_3	58	58	377	2128	0.470
F_4	43	43	91	416	2.404
F_5	51	51	114	232	4.310
F_6	55	59	99	140	7.143
F_7	64	72	186	1634	0.612
F_8	83	130	320	2886	0.347
F_9	160	160	667	16806	0.060
F^2_1	98	295	—	—	3.390
F^2_2	66	66	—	—	15.15
F^2_3	75	137	—	—	7.299
F^2_4	51	51	—	—	19.61
F^2_5	50	50	—	—	20.00
F^2_6	52	57	—	—	17.54
F^2_7	57	69	—	—	14.49
F^2_8	79	171	—	—	5.848
F^2_9	70	155	—	—	6.452

Table 2: Time (in msec) using a single renderer.

An important point to notice is that triangles are displayed as soon as they are available on the visualizer; the user does not need to wait for all the visible geometry to be sent from the SMA. Although frames 2 and 7 require 3.1 and 1.6 sec to be completely available, most of their triangles are visualized in less than 72 msec. For frame 2, 2209 triangles are stored on the visualizer data structure (they were required for frame1) and can be visualized immediately; 312 triangles are retrieved from the SMA cache and are visualized 58 msec after changing the view point. The user only needs to wait for 1210 triangles, less than one third, to visualize all the geometry. For frame 7, 3340 triangles out of 3732 are visualized after 72 msec. In a walkthrough the user can change the view point whenever he wants and does not need to wait for all the geometry to arrive; this ability allows him to navigate through the model with lower quality images at an acceptable frame rate (about 10 fps), stopping and waiting whenever he reaches a point of interest.

Frame 1 requires 39 sec for all the geometry to be available. This is a worst case: since it is the first frame on the walkthrough both the SMA cache and visualizer data structure are empty, all triangles vertices have to be shaded by the renderer. But the visualization is progressive and the user has feedback after 426 msec. Figure 3 shows frame 1 after 5 and 10 sec; although 5 sec are not enough to get a good perception of the whole image (538 rendered triangles), 10 sec are clearly enough (1379 triangles), specially if the user wants to navigate through the virtual model.

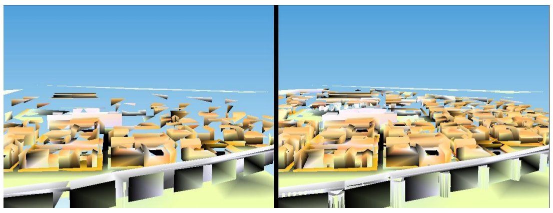

Figure 3: Frame 1 after 5 and 10 seconds.

The same frame ray traced with one primary ray per pixel (no antialiasing) with Radiance's rpict has a better quality, as shown on the right hand side of figure 4. However, it took 15% longer than the sparse sampled frame (first traverse) and two orders of magnitude slower on the second pass, even considering that Radiance caches irradiance values. Progressive refinement can improve the perception of faster high quality image generation. Also, tone mapping on the current version is clearly unable to convey the same image quality as Radiance's tone mapping algorithm.

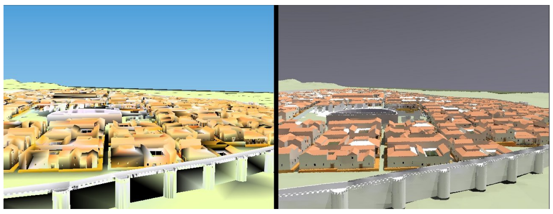

Figure 4: Frame 1 rendered using sparse sampling and one primary ray per pixel.

Since rendering is the most demanding task of the whole system, and since ray tracing is embarrassingly parallel, parallel computing is an obvious approach to increase performance. We rewrote Radiance to run on a cluster of workstations using MPI as the communication layer. Radiance is not embarrassingly parallel due to the irradiance cache, which is a spatially shared data structure. Sharing is achieved using the Network File System [War94]. Table 3 presents the rendering times for all the visible geometry obtained using 1 to 4 rendering processes, without any concern with parallel optimization. These results suffer from two severe penalties. The virtual model requires 1.1 GB RAM, but only 1 GB is available at the additional rendering nodes. The operating system will thus resort to virtual memory, which will decrease CPU utilization. Furthermore, the irradiance cache is shared using NFS over the 100 Mbit/s Ethernet; this bottleneck can be removed by storing this data structure on central