Exploring Virtual Prototypes Using Time-Critical Multiresolution Rendering

Enrico Gobbetti, Riccardo Scateni, and Marco Agus

CRS4 - Center for Advanced Studies, Research and Development in Sardinia
VI Strada Ovest, Z.I. Macchiareddu, C.P. 94, I-09010 Uta (CA), Italy,
E-mail: {Enrico.Gobbetti,Riccardo.Scateni,Marco.Agus}@crs4.it
WWW: http://www.crs4.it

Abstract. We present an application of our time-critical multiresolution rendering algorithm to the visual and possibly collaborative exploration of large digital mock-ups. Our technique relies upon a scene description in which objects are represented as multiresolution meshes. We perform a constrained optimization at each frame to choose the resolution of each potentially visible object that generates the best quality image while meeting timing constraints. We present numerical and pictorial results of the experiments performed that support our claim that we can maintain a fixed frame-rate even when rendering very large datasets on low-end graphics PCs.

1 Introduction

When undertaking a large and long-time lasting engineering or architectural project, it is vital to verify quite often what could be the consequences of the decisions taken during the design phase. Nowadays this is usually done by crafting physical mock-ups, typically made of wood or plaster, that help the designers to visualize the final result. Moreover, mock-ups are routinely used for applications such as testing equipment integration, accessibility and space requirements in domains ranging from aerospace and automotive manufacturing to architecture.

The aim of virtual prototyping research is to depart from this costly practice and allow architects, engineers and designers to work on digital mock-ups which simulate visual appearance and behavior of objects on a computer [18]. The expected benefits of virtual prototyping technology include:

- a substantial reduction of development time and of manufacturing costs, thanks to a reduced need for expensive physical mock-ups;
- the ability to continuously maintain digital mock-ups in sync with the design, and therefore the possibility to use them for documentation purposes and as a basis to help the dialogue between engineers from different fields who can talk, possibly with different words, about the same thing;
- the possibility of using mock-ups during collaborative design sessions among geographically distant partners, a very difficult option when using physical prototypes.

As for all virtual reality applications, virtual prototyping system have very stringent performance requirements: low visual feedback bandwidth can destroy the illusion of animation, while high latency can induce simulation sickness and loss of feeling of control. As virtual prototyping tools have to deal with very large dynamic graphics scenes with a complex geometric description, rendering speed is often the major bottleneck.

In addition, models are typically the output of a CAD elaboration, and thus are described in terms of curves and surfaces or as CSG models. Since current graphics boards provide adequate real-time performance only when drawing polygon meshes, a tessellation phase is always needed. Typical virtual prototypes, once converted to an adequate accuracy often exceed the millions of polygons and hundreds of objects, which poses important challenges to application developers both in terms of memory and speed.

As the complexity of a scene visible from a specific view-point is potentially unbound, meeting the performance requirements dictated by the human perceptual system requires the ability to trade rendering quality with speed. Ideally, this time/quality conflict should be handled with adaptive techniques, to cope with a wide range of viewing conditions while avoiding worst-case assumptions. The presence of moving parts, and the need for interaction of virtual prototyping tools, limits the amount of precomputation possible, leading to run-time solution. The traditional approach to render these scenes in a time-critical setting is to pre-compute a small number of independent level-of-detail (LOD) representations of each object composing the scene, and to switch at run-time between the LODs. This technique has multiple drawbacks, both in terms of memory requirements and quality of results. We recently demonstrated that these problems are overcome when using appropriate multiresolution data structures which enable to express predictive LOD selection in the framework of continuous convex constrained optimization [9, 8]. In this paper, we present an application of these techniques to the rendering of digital mockups. This research has partially been developed in the frame of a European Esprit project focusing on the development of new systems for computer supported collaborative work. We experimented with the system using the digital model of a very large machine that is under construction at CERN in Geneva that, once built, will be the largest machine for High Energy Physics in the world [3].

The rest of the paper is organized as follows. In section 2 we introduce the problem of rendering complex geometric scenes within a time budget, in section 3 we briefly describe our proposed solution and in section 4 we analyze the results obtained.

2 Time-critical rendering

Explorations in virtual environments require an high degree of interactivity, low-latency, and real-time processing capabilities while ensuring an high and steady frame-rate. The visual complexity of these environments has high variability and is potentially unbounded. Given that the rendering capabilities of whatever computer are instead limited, the only possible solution pass trough the usage of algorithms and systems for time-critical rendering, that are able to keep a constant frame-rate trading rendering quality with speed. The time-critical rendering problem is an optimization problem consisting in rendering the best quality image within timing constraint.

A time-critical rendering system is inherently a complex system that should combine algorithms such optimization algorithms, like ours, with occlusion culling [21], image based rendering [2] and other rendering acceleration methods [1]. Since the description of such a general framework is beyond the scope of this paper we give only a brief description of the background of our work.

2.1 Levels-of-details

The most straightforward way to store and retrieve multiple resolutions of a single geometric mesh is the generation of a fixed number of independent resolutions from the original mesh. Each of these simplified meshes retains as much as possible the topological and geometric characteristics while reducing number of vertices and triangles and thus the level of detail of the representation.

Many applications dealing with time-critical graphics (e.g. OpenInventor [14] and VRML [19]) include the possibility to store 3D models in levels-of-details (LODs). The main advantages of LODs are the simplicity of implementation and the fact that, as LODs are pre-calculated, the polygons can be organized in the most efficient way (triangle strips, display list), exploiting raw graphics processing speed with retained-mode graphics. The main drawbacks of this technique are related to its memory overhead,which severely limits in practice the number of LODs per object. As an example, representing an object at just the four LODs 100% (original mesh), 75%, 50%, 25% would cause an overhead of 150% over the single resolution version, which is an important drwaback for large virtual prototypes. The small number of LODs might introduce visual artifacts due to the sudden changes of resolution between differing representations [10] and, more importantly, limits the degrees of freedom of the LOD selection algorithm.

A different approach relies upon a scene description in which objects are represented as multiresolution triangle meshes, that is compact data structures able to efficiently provide on demand a triangle mesh approximation of an object with the requested number of faces [15]. In other words the number of precomputed levels of details of the original mesh is equivalent to the number of triangles describing it. This is the representation we choose for storing the models in our system.

2.2 Dynamic simplification

An alternative to per-object LOD selection is to dynamically re-tessellate visible objects continuously as the viewing position shifts. As dynamic re-tessellation adds a run-time overhead, this approach is suitable when dealing with very large objects or static environments, when the time gained because of the better simplification is larger than the additional time spent in the selection algorithm.

For this reason, this technique has been applied when the entire scene, or most of it, can be seen as a single multiresolution object from which to extract variable resolution representations. To support interactive animated environments composed of many objects, this paper focuses on per-object view-independent resolution rendering.

2.3 Resolution selection

The core of the optimization problem is choosing, for each object in the scene, at each frame, the most adequate resolution. The algorithm adopted must be able to choose the representation that gives the best visual clues to the user while keeping the overall rendering time under the budget given.

Run-time LOD selection is typically done using static heuristics or feedback algorithms. Static heuristics (e.g. distance-based [19], coverage-based [14], or perceptually motivated [16]LOD mappings) are not adaptive and are therefore inherently unable to produce uniform frame rates, while feedback algorithms, which adaptively vary LOD mappings based on past rendering times (e.g. [17]) suffer of unavoidable overshoot and oscillation problems when the complexity of the environment varies rapidly: entering in or exiting from a room in a walkthrough application is a typical example.

As demonstrated by Funkhouser and Séquin [6], to guarantee bounded frame times, predictive algorithms that optimize LOD selection based on estimates of rendering time and image degradation must be used. Having a *guarantee* on the total maximum lag of the application is a necessary precondition for using prediction techniques for lag reduction [20]. Unfortunately, the combinatorial optimization problem associated to LOD selection is equivalent to a version of the Multiple Choice Knapsack Problem [6, 13], which is NP-complete, and thus approximation algorithms have to be used. Current state-of-the-art techniques (Funkhouser and Séquin's greedy algorithm [6] and Mason and Blake's [13] incremental technique) produce a result which is only guaranteed to be half as good as the optimal solution and have a running time depending both on the number of objects and on the number of LODs per object.

Since we use multiresolution triangle meshes, we can approximately describe the representation as continuous instead of discrete and thus elaborate continuous a convex constrained optimization strategy [9, 8].

3 Our approach

As we said, our approach relies upon a scene description in which objects are represented as multiresolution triangle meshes. At each frame, we aim to find within a fixed short time the triangle mesh representation for each potentially visible object that produces the "best quality" image within the target frame time. This is done in an optimization phase which takes as input the set of potentially visible objects determined by a culling algorithm (e.g. bounding box or occlusion culling) and selects as output the list of triangle meshes to be rendered.

3.1 Optimization

To perform the optimization task we define the two functions $cost(\mathcal{W}, S(r))$ and $degradation(\mathcal{W}, S(r))$. The $cost(\mathcal{W}, S(r))$ heuristic provides an estimation of the time necessary to render in a viewing configuration $\mathcal{W}$ (camera, lights), a scene composed of the multiresolution objects present in S at resolutions r. The $degradation(\mathcal{W}, S(r))$ heuristic provide an estimation of the perceptual distance between the image produced by rendering in a viewing configuration $\mathcal{W}$ a scene composed of the multiresolution objects present in S at resolutions r and the image obtained in the same configuration with all objects at full resolution.

Using this formalism, our approach to predictive adaptive rendering is stated as follows:

$$\begin{aligned} \text{Minimize:} \quad & degradation(\mathcal{W}, S(\mathbf{r})) \\ \text{Subject to:} \quad & cost(\mathcal{W}, S(\mathbf{r})) \leq t^{(\text{desired})} \\ & r \succeq \mathbf{r}_m in \\ & r \preceq \mathbf{1} \end{aligned} \tag{1}$$

where $\succeq$ and $\preceq$ denote componentwise inequality, $r_m in$ is the vector containing lower bounds for the resolution of each object, and $t^{(\text{desired})}$ is the target display time.

The difficulty of solving problem 1 depends largely on the nature of the *degradation* and *cost* heuristics. We have demonstrated that simple forms of these heuristics can be found, leading to efficient solution methods applicable in a time-critical setting [8]. We make the simplifying assumption that object quality depends exponentially on the resolution by a law of the form $w_{interest} r^{-\alpha}$ for some algorithm parameter $\alpha > -1$, and that frame time depends linearly on the resolution r for $r > r_{min}$, where r_{min} is determined by finding the minimal resolution at which the graphics pipe-line bottleneck is not rasterization. It is possible to show that in this case problem 1 can be solved very efficiently using an active-set method [5, 7], the non-linear equivalent of the well-known simplex linear optimization method. The performance of the method depends on how efficient it is to generate an initial feasible solution, to solve equality constrained problems during the iterative refinement phase, and to compute Lagrange multipliers for checking convergence. All these sub-problems can be solved in linear time in the case of resolution optimization. For details, refer to the original publication [8].

3.2 Multiresolution Data Structure

Our optimization algorithm is independent from the particular data structure used to represent multi-resolution meshes. The only requirements are the ability to represent a mesh with an arbitrary number of triangles and to traverse the structure at arbitrary resolutions faster than the graphics pipe-line.

Nevertheless along with implementing the optimization and rendering algorithm we developed a companion data structure (TOM, Totally Ordered Mesh) satisfacting these requirements [9]. As we are dealing with large datasets, we have focused on devising a representation with a small memory footprint. The construction of TOM is based on a single, simple operation: the vertex pair contraction operation, denoted $(v_1, v_2) \rightarrow \tilde{v}$, that replaces two vertices (v_1, v_2) with a single target point $\tilde{v}$ to which all the incident edges are linked, and removes any triangles that have degenerated into lines or points.

As in [12], we implement mesh simplification schemes based on iterative vertex substitution in a generic framework of greedy algorithms for heuristic optimization. The generic greedy algorithm is parameterized by a *binary oracle*, deciding which vertex substitutions are legal, and a *fairness predicate*, which assigns priorities to all vertex substitutions in the legal candidate sets. By iteratively applying this operation a triangle mesh can be reduced by removing one vertex and possibly some degenerated faces at a time. As in [4, 12] we have noticed in our various experiments that the most important factor to preserve quality during mesh simplification is the order in which operations are done, and that a good mesh reduction scheme can be achieved without inserting new vertices.

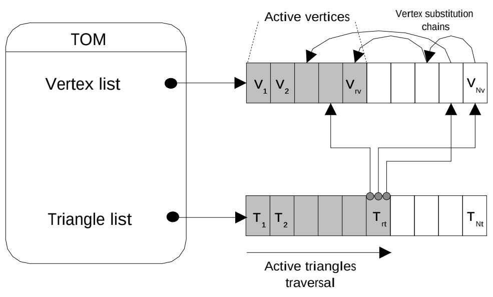
TOM
Vertex list
Triangle list
Active vertices
Vertex substitution chains
V_1 V_2 V_{rv} V_{Nv}
T_1 T_2 T_{rt} T_{Nt}
Active triangles traversal

Fig. 1. TOM data structure. Multiresolution meshes are stored using a vertex list and a triangle list sorted according to contraction resolution.

On the multiresolution mesh we can define a total order on both the list of vertices and the list of triangles. Sorting according to this order after the simplification generates a compact and efficient data structure (see figure 1). By ordering the vertex list, we obtain a packed representation where the active vertices at vertex resolution $r_v = \frac{n}{N_v}$ are exactly the first n ones in the vertex array of size N_v (fig. 1). Moreover, by ordering the triangle list, we have a way to iterate through the triangles that define the mesh at an arbitrary resolution in a time depending only on the number of active triangles and the lengths of the vertex substitution chains. The memory overhead introduced is limited to the space required to store the vertex substitution history and is approximately 8%.

4 Results and discussion

The time-critical multiresolution scene rendering algorithm briefly described in this paper has been implemented and tested on Silicon Graphics IRIX and Windows NT machines. The TOM data structure is integrated in the CAVALCADE collaborative virtual prototyping system [18].

In this section we report and analyze some results of a sample session of visual exploration of a large dataset composed of more than 300 separate parts and more than 1 million triangles overall. The results presented here were obtained on a Silicon Graphics 320 PC running Windows NT 4.0 and configured with a single 500 MHz Pentium III processor with 512 Kb L2 cache, 512 Mb RAM, and a Cobalt graphics chipset.

The dataset represents the ATLAS Experiment Pit of the LHC (Large Hadron Collider), a machine under construction at CERN in Geneva. It was obtained first transforming the original Euclid CSG CAD model in VRML using an accurate tessellation, then processing it to generate the TOM representation. The LHC is a large scale project, where the design phase is probably the most delicate one: this is the moment when some critical choices are taken which might dramatically affect the final results, timing and costs. The possibility to interactively manipulate the digital model is essential to a good understanding of the interrelationships between the parts. It is nowadays impossible to rely only on the visual capabilities of the present CAD tools, and a system like ours can thus be a very useful supplement for the design phase.

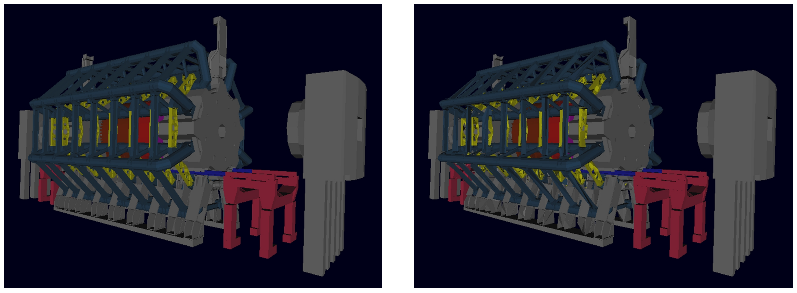

Fig. 2. A view of the whole ATLAS Experiment Pit. Comparison between full (left, 1,072,858 triangles) and adaptive resolution (right, 41,487 triangles).

4.1 Memory needs

The entire multiresolution dataset occupies 26.2 Mb using a TOM representation for each of the objects, considering 32 bits to store both integer and floating

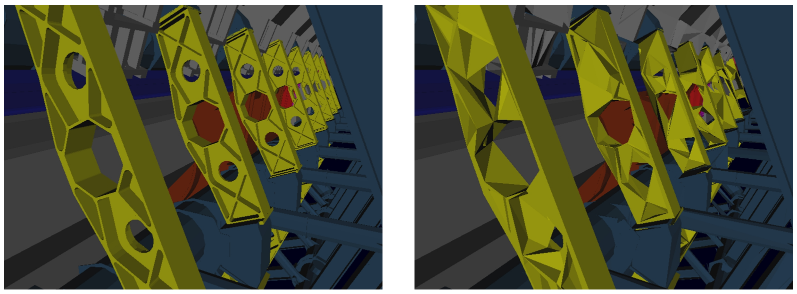

Fig. 3. Inside the ATLAS Experiment Pit. Comparison between full (left, 1,035,533 triangles) and adaptive resolution (right, 43,989 triangles).

point data and storing only normals as vertex attributes. The same dataset in VRML face-vertex form (single resolution) occupies 24.2 Mb, that is only 8% less than the multiresolution version. By constrast, when using a VRML scene graph with six levels of details per object (100%, 50%, 25%, 12%, 6%, 3%), storage requirements grow to 47.6 Mb, introducing an overhead of 96%. Memory needs for the PM representation range from 28.3 Mb to 64.7 Mb, i.e 17% to 67% more than the single resolution version, depending on the resolution at which objects have to be rendered [11]. The compactness of TOM is very important for large virtual prototypes.

4.2 Walkthru

We have simulated in a short recorded session of less than one minute several typical possible ways of interacting with the huge model: first the observer has an overview of the whole model from different viewpoints; after this complete loop around the machine the observer decide to have a closer look at a particular section attracting her interest; she then moves to look at another piece of the machine and rapidly turns back to inspect a particular component along the previous path.

The picture on the right of figure 2 represents the scene as seen by the observer when looking at the whole model. The picture on the left (taken at full resolution) allow to appreciate the slight degradation when passing from the original to the multiresolution model. It is interesting to point out that, while with our system we are able to keep a steady 10 frames per second rate, rendering the full model on the same machine requires an average of 1 frame per second.

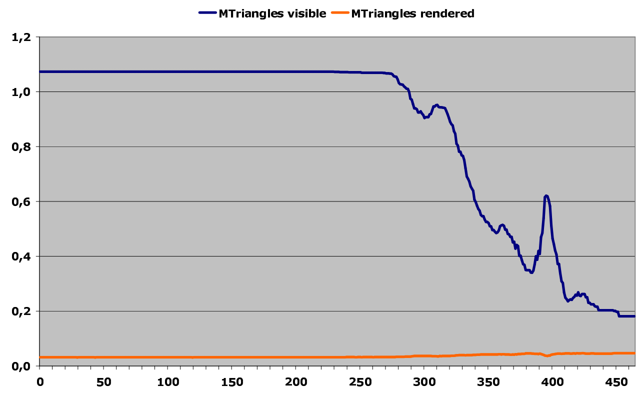

Fig. 4. Triangles visible and rendered for each frame. For each observer viewpoint along the visual exploration path the number of triangles in the view frustum (visible) and the number of triangles actually rendered are reported.

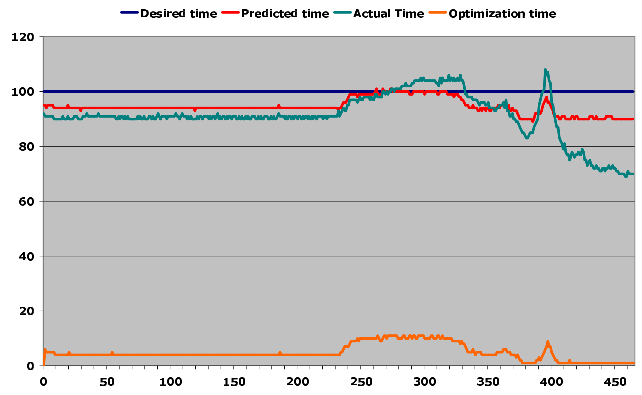

Fig. 5. Time report for each frame. For each rendered frame four are reported parameters.

In figure 3 there is a representation of the scene seen by the observer when *entering*into the dataset. Here we can see that the visual degradation of some of the structures represented is higher, but for the sake of the investigation the quality is more than acceptable.

In figure 4 we plotted, for each frame, the number of triangles potentially visible by the observer and the number of triangles rendered. The system does not perform any back face or occlusion control, so the actual number of visible triangles can be even much less than the number reported. It is interesting to notice that, as one can expect, the number of rendered triangles is quite steady, despite the complexity of the model in the view frustum. This is an evident performance index for both the optimization strategy and the multiresolution data structure. The triangle budget is not constant, due to variations caused by fill-rate, which are particularly important for low-end machines.

In figure 5 we plotted indicators that show how the system performs under different situations. The desired time for each frame is fixed to 100 ms (giving a 10 f/s rate), the predicted time is deducted from the estimation provided by the *cost* heuristic, the optimization time and the actual total rendering time are measured. As we can see, there is a good agreement between predicted and actual time, which are both maintained close to the target. The oscillation in rendering time depend mostly on the time taken by the optimization routine, which often finishes before the user-imposed limit of $10ms$. When this happens, the frame quality is optimal according to the heuristics. In all other cases, iterative refinement is terminated earlier, and a sub-optimal solution is used. The system is anyway robust enough to absorb these changes without any evident visual oscillation.

5 Conclusions and future work

We presented here a study of an application of a multiresolution based time-critical system to the visual exploration of a large engineering model that is impossible to examine at interactive speed even on high-end graphics workstations. The results we reported allow us to say that our system can be useful when designing such large machines to perform visual analysis at early stage of the design phase. The system enables the handling of scenes totaling millions of polygons and hundreds of independent objects on a standard graphics PC. The technique does not rely on visibility preprocessing and can be readily employed on animated scenes and interactive VR applications.

We plan to perform similar experiments with architectural datasets, where our system can be used to walk through the digital mock-up even during the early stages of conceptual design. We are working on extending the data structure to support high-speed variable resolution traversals. In this case, the optimization algorithm will assign a polygon budget to each of the visible objects based on timing constraints, but each object will autonously decide how to distribute this budget at each frame. This approach will improve the visual quality of scenes composed of large objects with linked pieces.

Acknowledgments

The ATLAS dataset is courtesy of CERN, Geneva, Switzerland. This research is partially sponsored by the European project CAVALCADE (ESPRIT contract 26285). We also acknowledge the contribution of Sardinian regional authorities.

References

1. D. Aliaga, J. Cohen, A. Wilson, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, E. Baker, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. A framework for the real-time walkthrough of massive models. Technical Report TR98-013, Department of Computer Science, University of North Carolina - Chapel Hill, Mar. 26 1998.
2. D. Aliaga and A. Lastra. Automatic image placement to provide a guaranteed frame rate. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, Aug. 1999.
3. J.-F. Balaguer. VRML for LHC engineering. In M. Goebel, J. David, P. Slavik, and J. van Wijk, editors, *Virtual Environments and Scientific Visualization '96*, pages 159–168. Eurographics, Springer, 1996.
4. S. Campagna, L. Kobbelt, and H.-P. Seidel. Efficient decimation of complex triangle meshes. Technical Report 3, Computer Graphics Group, University of Erlangen, Germany, 1998.
5. R. Fletcher. *Practical Methods of Optimization. Volume 2: Constrained Optimization.* Wiley, New York, NY, USA, 1981.
6. T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics (SIGGRAPH '93 Proc.)*, 1993.
7. P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, 1981.
8. E. Gobbetti and E. Bouvier. Time-critical multiresolution rendering of large complex models. *Journal Computer-Aided Design (to appear).*
9. E. Gobbetti and E. Bouvier. Time-critical multiresolution scene rendering. In D. Ebert, M. Gross, and B. Hamann, editors, *IEEE Visualization '99*, pages 123–130. IEEE Computer Society, ACM Press, 1999.
10. P. S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, Banff, Canada, May 1994. Canadian Inf. Proc. Soc. URL: `http://www.cs.cmu.edu/~ph`.
11. H. Hoppe. Efficient implementation of progressive meshes. *Computers and Graphics*, 22(1):27–36, January 1998.
12. L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In W. Davis, K. Booth, and A. Fourier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 43–50, San Francisco, June18–20 1998. Morgan Kaufmann Publishers.
13. A. E. W. Mason and E. H. Blake. Automatic hierarchical level of detail optimization in computer animation. *Computer Graphics Forum*, 16(3):191–200, Aug. 1997. Proceedings of Eurographics '97. ISSN 1067-7055.
14. Open Inventor Architecture Group. *Open Inventor C++ Reference Manual: The Official Reference Document for Open Systems.* Addison-Wesley, Reading, MA, USA, 1994.

15. E. Puppo and R. Scopigno. Simplification, LOD, and multiresolution: Principles and practice. Eurographics tutorial notes, 1997.
16. M. Reddy, B. Watson, N. Walker, and L. F. Hodges. Managing level of detail in virtual environments: a perceptual framework. *Presence: Teleoperators and Virtual Environments*, 6(6):658–666, 1997.
17. J. Rohlf and J. Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In A. Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
18. P. Torguet, O. Balet, E. Gobbetti, J.-P. Jessel, J. Duchon, and E. Bouvier. Cavalcade: A system for collaborative prototyping. In *Proc. International Scientific Workshop on Virtual Reality and Prototyping*, May 1999. Conference held in Laval, France, June 3-4.
19. VRML 97, International Specification ISO/IEC IS 14772-1, Dec. 1997.
20. M. Wloka. Lag in multiprocessor virtual reality. *Presence: Teleoperators and Virtual Environments*, 4(1):50–63, Sept. 1995.
21. H. Zhang, D. Manocha, T. Hudson, and K. E. Hoff III. Visibility culling using hierarchical occlusion maps. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 77–88. ACM SIGGRAPH, Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.