A QoS FRAMEWORK FOR INTERACTIVE 3D APPLICATIONS

N. Pham Ngoc[1], W. van Raemdonck[2], G. Lafruit[2], G. Deconinck[1], and R. Lauwereins[2]

[1]Katholieke Universiteit Leuven-ESAT/ACCA
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium
Email: Nam.Phamngoc@esat.kuleuven.ac.be

[2] IMEC-DESICS, Kapeldreef 75, B-3001 Leuven-Heverlee, Belgium

ABSTRACT

We present a QoS (Quality of Service) framework for interactive 3D applications, in which the QoS management relies on high-level QoS parameters of quality scalable 3D objects, which are transmitted from the service provider to the user terminals. PSNR is used as one of these high level parameters for representing the perception quality of 3D objects. In real-time, interactive 3D applications, one of the tasks of the QoS management at the end-user's terminal consists in keeping a constant interactive frame rate, by trading off the 3D scene perception quality for frame rate, under resource constraints. We show that this task involves solving a NP-hard optimization problem and we present an approximation algorithm that solves the problem with an accuracy of more than 95%, compared to the optimal solution, while representing a negligible computation effort for every frame. Experimental results show the soundness of the proposed framework and algorithm.

Keywords: 3D Quality of Service, level of detail, interactive applications, optimization, PSNR

1. INRODUCTION

Currently, with the increasing processor performance and network bandwidth, many interactive virtual reality applications such as 3D games, virtual museum[1] or virtual shop applications have become feasible on a wide range of platforms. One important requirement of these applications is that the interactive frame rate should be high enough and should be constant in order to give the user a smooth navigation in the virtual environment. However, due to the heterogeneity in end-system processing capacities and the un-bounded complexity of 3D contents, this requirement can hardly be met without adapting the applications to the processing power of the user terminal. There have been a number of adaptation techniques such as visibility processing and level of detail selection [Schma97a], which have different efficiency under different assumptions.

In this paper we present a QoS framework for guaranteeing the user specified interactive frame rate by degrading the quality of 3D objects in such a way that a minimal overall quality degradation over the scene is obtained. This process involves solving a NP-hard optimization problem at every frame. In the framework, the management of QoS at end-user systems relies on high-level QoS parameters of quality scalable 3D objects, which are transmitted from the service provider to the user terminals. In addition, we consider only those kinds of scalable mesh objects that can be simplified non-uniformly in order to preserve as much as possible the object's silhouette [Benic99, Sande2000, Cohen98, Hoppe97, Hoppe98, Luebk97]. With this assumption, the quality degradation can be approximated by the PSNR between the object's image, rendered at full mesh resolution and the object's 2D image obtained after rendering with degraded settings (less mesh resolution). We also propose in the framework an approximation algorithm as a solution for the NP-hard problem that yields an approximated solution with a high accuracy, within a very short time.

[1] See http://www.fnmt.es/esp/museo/evisita.htm for an example of a virtual museum.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 describes the QoS framework including a QoS architecture, QoS and resources models. The optimization problem is also presented in this section. Section 4 presents the approximation algorithm for the optimization problem. Experimental results and discussion are given in section 5 and finally section 6 presents our conclusions and future work.

2. RELATED WORK

As respect to QoS at the end-systems, much effort has been spent on video domain [Pham00]. For example, Bril [Bril01] describes a QoS framework for adapting the quality of scalable videos to the processing capacity of the video decoder. In this paper, we follow the same lines of thought for scalable 3D objects. Earlier, Brandt [Brand98] proposed a simple QoS management framework, which guarantees QoS for a small number of 3D objects, in which each object has only a few quality levels. Our framework, on the contrary, deals with much more complex applications, having a large number of possible quality settings and objects.

Funkhouser and Sequin [Funkh93] were among the first to propose a predictive algorithm that optimizes LOD selection based on benefit and cost heuristics to guarantee bounded frame times. They use a well-known greedy algorithm to find an approximated solution for the optimization problem which has an accuracy of 50% as compared to the optimal solution. Gobbetti [Gobbe99] adopts the same approach and extends it with a more accurate algorithm with a restriction that benefit and cost functions must be convex and smooth, which is not always valid (e.g. the quality of a 3D object with different rendering modes corresponds to rather a step benefit function than a smooth function). Both approaches use the *accuracy* factor in the benefit function to measure how well the mesh at a certain resolution approximates the mesh at maximum resolution. The accuracy factor was chosen in both approaches as a function of the number of polygons, which is independent of the viewpoint and the quality of the texture. Our PSNR parameter, although it is not a perfect quality measure, can express the dependency of the quality of 3D objects at a certain mesh resolution with the viewpoint and the quality of the texture. PSNR is therefore clearly a better quality measure. The approximation algorithm we propose works on discrete benefit/cost points and has an accuracy of more than 95%, which is defined as the ratio of the total benefit of the approximated solution over the benefit of the optimal solution.

3. QoS FRAMEWORK

3.1QoS architecture

Our QoS architecture is shown in Fig. 1. In order to support QoS at the end user terminal, the service provider (e.g. a web server) should negotiate a QoS contract with the network and with the user terminal. It therefore uses (i) a 3D objects database, in which scalable 3D objects are encoded, (ii) a QoS profile database in which high level QoS parameters of 3D objects are stored and finally (iii) a manager that manages the databases. At the user terminal, the QoS architecture consists of a QoS negotiator, a user interface, a QoS manager and a resource estimator. The QoS negotiator has the same function as its counterpart at the service provider side. The user interface allows the user to specify a desired interactive frame rate and to interact with the 3D scene. The resource estimator estimates the processing time for each object based on high-level QoS parameters of the object and platform specific parameters. The function of the QoS manager is to maintain a constant specified interactive frame rate by executing the control loop of Fig. 2, i.e. selecting for each viewpoint the best quality parameters of the visible object portions, taking the constraints on the resources into account.

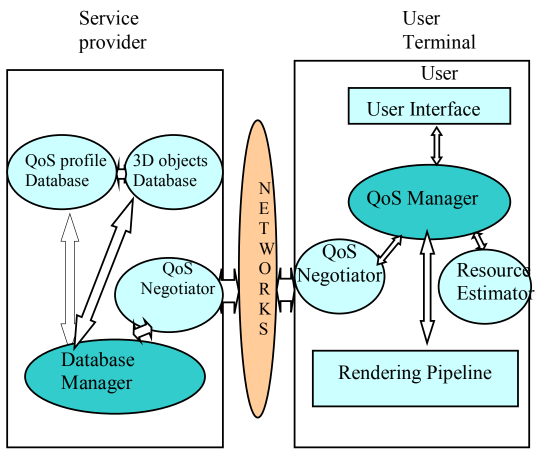

Figure 1. QoS Architecture

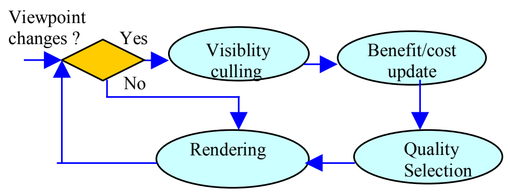

Figure 2. Frame rate control loop

3.2 QoS model

3.2.1 PSNR as a quality metric

It is clear that the quality of a 3D object depends on the mesh resolution (or level of detail) and on the rendering mode (flat shading, smooth shading or texture rendering). For texture rendering, the quality of the 3D object depends also on the quality of the texture. We observe that given a mesh simplification algorithm that simplifies the mesh at a lower resolution, the quality degradation is different for each viewpoint . In addition, at a certain viewpoint, different mesh resolutions may give the same observed quality degradation, thanks to the masking effect of the texture. Therefore, modeling the quality degradation of 3D objects by a simple function of the number of polygons as in [Funkh93] and [Gobbe99] is not appropriate. Instead, we use PSNR, a well-known quality measure in the 2D image processing domain, as a quality metric to approximate the quality degradation between the object's image, rendered at full mesh resolution and the object's 2D image obtained after rendering with degraded mesh resolution. By using PSNR, the dependency of quality degradation on texture quality can also be modeled. Note that at the current state of the framework, we consider only texture rendering and the quality of the texture is fixed. This assumption will be released in future work where we will consider texture degradation mechanisms. The quality degradation of 3D objects therefore relies solely on changing mesh resolution. To represent the viewpoint dependency of quality degradation, we propose that at each mesh resolution, PSNRs are measured in advance for a set of viewpoints around the object, from which the PSNR for any other viewpoint can be estimated by barycentric interpolation.

3.2.2 Other high level QoS parameters

Besides providing PSNRs, the service provider should also provide for each object the following high level QoS parameters: (i)the semantic relative importance β(O) of the object in the scene (e.g. in a virtual shop the 3D model of a product is more important than the 3D model of the wall of the shop), (ii) the number P of polygons for each quality level and (iii) a set of number S of projected pixels for some viewpoints around the object. Similar to PSNR, the value S at any other viewpoint can be estimated by barycentric interpolation. P and S will be used by the resource estimator to estimate the rendering time of the object as will be explained in section 3.3. S will also be used by the QoS manager as one factor to calculate the benefit value as will be explained in section 3.2.3. All these high level parameters may be stored in a table in the QoS profile database, as given by the example of Table 1.

Table 1. QoS profile

Level	Object name: chair β=0.5				
	Viewpoint	Vp1	Vp2	....	VpN
	S (pixels)	100000	200000	…	50000
	#Polygons	PSNR	PSNR		PSNR
1	200	30.5	34		35
2	400	34	37		40
...					
L	2000	47	50.7		47

3.2.3 View-dependent benefit function

Denote B(O, L, V) to be the benefit of object O, rendered at quality level L and viewpoint V. The benefit value represents the amount of perception contribution of the object to the overall scene. It depends on the PSNR(O, L, V), the size of the object S(O) (in % screen coverage), the semantic meaning β(O) of the object in the scene and the viewing angle α from the user viewpoint to the object. In the following benefit function:

$$B(O,L,V)= \beta(O) *S(O)* \cos(\alpha)* PSNR(O,L,V)$$

β(O) is the only viewpoint independent parameter. The other factors have to be recalculated whenever the viewpoint changes (*benefit/cost update* step in Fig. 2).

3.3 Resource model

In order to estimate accurately the time needed to render an object at a certain quality level for quality–frame rate trade off, it is important to have an accurate resource consumption (i.e. CPU time) model. For this purpose we have profiled Mesa3D - a widespread implementation of OpenGL - to the number of OpenGL primitive function calls and their corresponding data accesses [Lafru98, Lafru00]. We conclude that the execution time of the 3D rendering engine is related to the number P of polygons and the number S of projected pixels. For low and moderate-end platforms without hardware accelerated cards, parameter S has a large impact on the total rendering time. However, for powerful 3D graphics hardware accelerated platforms, the number S of projected pixels has little impact on the execution time, since the bottleneck is located around the 3D mesh processing, rather than in the texture processing. Fig. 3 shows the relation between number of polygons and execution time for different rendering platforms with (HW) and without (SW) 3D graphics acceleration card. A dependency with the number *S* of projected pixels is observable when no

3D graphics acceleration card is used (SW). SW_y% corresponds to rendering with y% screen coverage. HW_y% is plotted “for any value of y”.

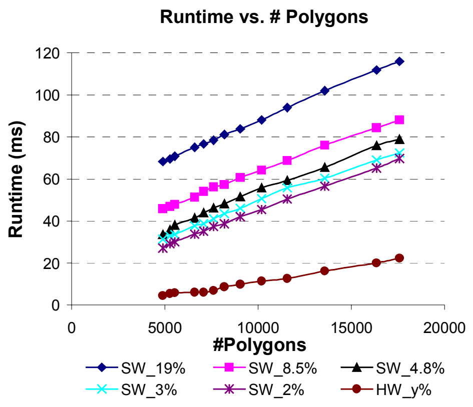

Figure 3.

Thus, if we let T(O, L, V) be the time needed to render object O at quality level L and at viewpoint V, the following relation holds:

T(O, L, V) = f (P, S), where function f is specific for each platform.

The resource estimator in our QoS architecture actually implements this function and therefore belongs to the platform dependent part of the architecture.

3.4 Optimization problem

The objective of the quality level selection step in the control loop in Fig. 2 is to select a combination of quality levels of all visible objects in the scene in such a way that the overall benefit of the scene is maximized while the total rendering time does not exceed the target frame time. Mathematically, the optimization problem can be stated as follows:

Maximize:

$$\sum_{i=1}^{N} \mathrm{B_i}(\mathrm{O_i}, \mathrm{L_i}, \mathrm{V_i})$$

Subject to:

$$\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{T_i}(\mathrm{O_i}, \mathrm{L_i}, \mathrm{V_i}) \leq \mathit{TargetFrameTime}$$

Where N is the number of visible objects at the viewpoint under consideration.

It can be easily seen that this optimization problem can be reduced to the 0-1 Knapsack problem and therefore is a NP-hard optimization problem for which an optimal solution cannot be found in real time [Garey79]. Heuristics are thus needed to find approximated solutions as fast as possible within an acceptable accuracy. The execution time of such an approximation algorithm should be much smaller, compared to the target frame time, in order to be used at every frame. In the following section, an approximation algorithm that satisfies this constraint is presented.

4. A SOLUTION FOR THE OPTIMIZATION PROBLEM

For each object O_i we define a list L[i] of (*benefit, cost*), i.e. (B, T), points. The first point in the list corresponds to the lowest quality level of object O_i and the last point in the list corresponds to the highest quality level. L[i] is thus a sorted list in the order of increasing rendering time (and also in the order of increasing benefit). The problem in section 3.4 is equivalent to finding a point in each list L[i] such that the overall benefit is maximized, while satisfying the timing constraint. Let:

- L[i].NrOfPoint be the number of point in list L[i]
- L[i][j] be the point j in list L[i]. L[i][j].level, L[i][j].benefit, L[i][j].cost be the quality level, the *benefit* value and the *rendering time* associated to L[i][j], respectively.
- (q[1], q[2],..., q[N]) be the solution vector. For example, the vector (2, 3, 5) means that object 1 can be rendered at quality level 2, object 2 at quality level 3 and object 3 at quality level 5.
- index[i] represents the current position of a point in list L[i].
- slack represents the remaining of rendering time that can be used to improve the quality levels of objects.

The pseudo code of the algorithm is given in Fig. 4. The algorithm is based on two heuristics, corresponding to two phases. In the first phase, a heuristic similar to the one in [Lee99] is used, which reduces the number of (*benefit, cost*) points in list L for each object to a smaller number of representative points in list L’. This is obtained by calling the RepresentativeList(L) function. This function implements an algorithm similar to Graham-Scan algorithm [Cormen90], which finds the convex hull of a set of points. In the second phase, we use the second heuristic, which is based on the *benefit/cost* ratio and a nice property of the list L’, i.e. L’ consists of non-decreasing slope segments (Fig. 5).

```
Algorithm (N, L[1], L[2],..,L[N])

1.  mincost=0;
2.  for i=1 to N do
3.      L'[i] = RepresentativeList(L[i]);
4.      index[i]=1;
5.      //Initialise the solution with the lowest quality
        ///for each object
6.      q[i]=L'[i][index[i]].level;
7.      mincost=mincost+L'[i][1].cost
8.  slack=targetframetime-mincost;

9.  //Now gradually improve the solution
    // with the improvement that corresponds to $\alpha_{max}$
10. While (true) do
11.   $\alpha_{max}$=0;
12.   for i=1 to N do
13.     If (index[i] < L'[i].NrOfPoint)
14.       j=index[i] +1;
15.       feasible=true;
16.       costDiff=L'[i][j].cost-L'[i][index[i]].cost;
17.       benefitDiff= L'[i][j].benefit-L'[i][index[i]].benefit;
18.       $\alpha$= benefitDiff/costDiff;
19.       if (L'[i][j].cost- L'[i][index[i]].cost > slack)
20.           feasible=false;

21.       if (feasible) and ($\alpha > \alpha_{max}$ )
22.          $\alpha_{max}$ =$\alpha$;
23.          i1=i; j1=j;
24.     if ($\alpha_{max}$=0)  break; //No more improvement
25. slack=slack-( L'[i1][j1].cost- L'[i1][index[i1]].cost);
26. q[i1]= L'[i1][j1].level;//Update quality level object i1
27. Return (q[1], q[2],...,q[N])
```

Figure 4.

The different steps of the algorithm are summarized as follows:

- The algorithm takes N lists L[1] to L[N] of (*benefit, cost*) points corresponding to N visible objects as the inputs.
- The function RepresentativeList (L) is applied for each list L[i] to get the list L'[i].
- The algorithm starts from the lowest quality for each object (i.e. the first point of L'[i]) and iterates to improve the overall quality until no more improvement can be obtained. At each iteration, only one object is selected and its quality level is improved such that the *benefit/cost* ratio is maximum. Fig. 5 shows that if the current quality level is at point A, we have 3 possibilities to improve the quality either to B, C or D. Since $\alpha 1 > \alpha 2 > \alpha 3$, going from A to B always gives us the highest *benefit/cost* ratio. Therefore, at each iteration, only the $\alpha 1$ of each object is calculated and compared to this value of other objects. The object with maximum $\alpha 1$ is selected and its quality level is improved by going from the current point in list L' to the next point in the list. This is a very important observation that helps to speed up the algorithm.

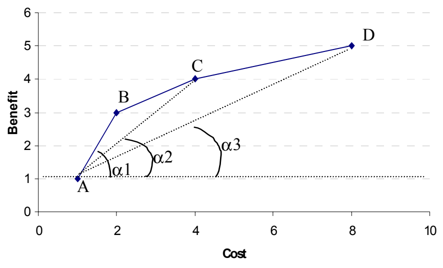

Figure 5.

The worst case computational complexity of the algorithm is O (N.L.logL) where L is the number of quality levels of the object that has the most quality levels.

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

Our implementation of the framework is written in C++ using OpenGL and GLUT libraries. The experiments presented in this section have been carried out on a PIII 866 MHz computer with an Elsa Gladiac Ultra 64 MB AGP 3D card running WinNT. A 3D scene of our company's main auditorium with 78 objects and a total of 112000 polygons created in Visual 3D Studio Max was used.

5.1. Performance of the quality selection algorithm

To evaluate the run-time efficiency of the proposed algorithm in speed and accuracy, we have run a number of simulations on randomly created (benefit, cost) lists. These lists were created randomly but consistently, i.e. a higher benefit corresponds to a higher cost. An optimal algorithm was also implemented in order to measure the accuracy of the approximation solution as compared to the optimal one. Fig. 6 shows the runtime of the approximation algorithms when the number of objects N varies from 5 to 200, while the number of quality levels of all objects are fixed at 10 and 50. Each point in the graph is obtained by taking an average of 5 different sets of data. The accuracy we measured in most of the runs is better than 95%.

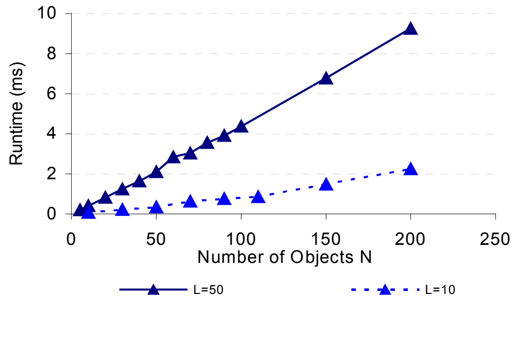

Figure 6.

5.2. Creation of QoS profile and resource model

We have developed a tool that reads a 3D scene which contains a lot of 3D objects, that are automatically transformed into scalable objects. A QoS profile is created for each object, similar to the example of table 1. The tool uses the mesh simplification algorithm of Garland [Garla97]. For each object in the auditorium model, we created on average 50 different quality levels. Fig. 7 shows the PSNR of the chair model at some different viewpoint as a function of the number of polygons. From this figure we clearly see the dependency of the quality degradation with the viewpoint as discussed in section 3.2.1. We also observe that at some points, the quality remains almost unchanged with an increase of the number of polygons. In this case, the quality selection algorithm will select the point with minimum number of polygons among all the points that have the same quality. If we had used the benefit function of [Funkh93] or [Gobbe99], the algorithm would have selected another, less optimal point.

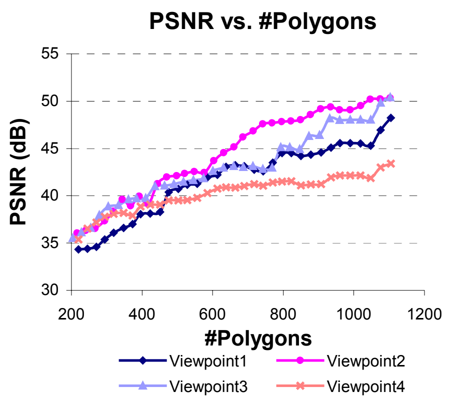

Figure 7. PSNR vs. #Polygons at different viewpoints

The resource model was obtained by profiling. For the test platform, the rendering time mainly varies as a linear function with the number of polygons.

5.3. Quality and frame rate trading off

To test the effect of trading off quality for frame rate on the visual quality of the whole scene, we kept the frame rate fixed and took the snapshot of the scene at several viewpoints. We then compared the quality of the scene with and without adaptive frame rate. Figure 8 shows the snapshots of the scene at two different viewpoints when the scene is rendered at the maximum quality (Fig. 8 a) and c)) and when it is rendered at a fixed target frame rate of 15 fps (Fig. 8 b) and d)). For the first viewpoint (Fig. 8 a) and b)), all 78 objects are visible and the frame rate obtained when the scene is rendered without quality degradation is 7 fps. For the second viewpoint, (Fig. 8 c) and d)), only 60 objects are visible and the frame rate obtained when the scene is rendered without quality degradation is 10 fps. The total time of visibility culling, benefit/cost updating and quality selections steps for both viewpoints introduced an overhead of less than 5 ms (i.e. approximately 10 % of the frame time budget).

In Fig. 8 b), we can see that degradation is easily observed at the chairs, the lamps and the loudspeakers. Although it is difficult to observe, the quality of the chairs in front was degraded more heavily than the quality of the chairs at the back (which are bigger and closer to the user viewpoint). Obviously, by applying an almost unnoticeable quality degradation, we obtain a fixed interactive frame rate. This fact proves the appropriateness of our QoS model.

6. CONCLUSIONS AND FUTURE WORK

We have presented a QoS framework that enables trading off quality for frame rate in order to have a smooth interaction in 3D interactive applications. In this paper, we have made three main contributions. Firstly, we have proposed a QoS architecture for 3D QoS and identified clearly the role of the service provider and of the user terminal in QoS specification and management of the overall QoS framework. Secondly, we have, for the first time, used PSNR as a quality metric to measure the quality degradation of 3D objects. It has been proved that, PSNR is better than other existing metrics for the same purpose. Thirdly, we have proposed a fast and accurate approximation algorithm for performing quality level selection, which can be used in a large range of emerging 3D interactive applications, more particularly in networked PC platforms. The experimental results have proved the soundness of our framework and algorithm.

Database management issue and QoS negotiation for minimizing the application set-up time have not been addressed and are subjects for future work. We will also consider the degradation of texture quality to introduce more freedom in the 3D QoS management.

ACKNOWLEDGEMENT

Part of the work presented in this paper has been done in the ITEA-EUROPA project. Geert Deconinck is a postdoctoral fellow of the Fund for Scientific Research-Flanders.

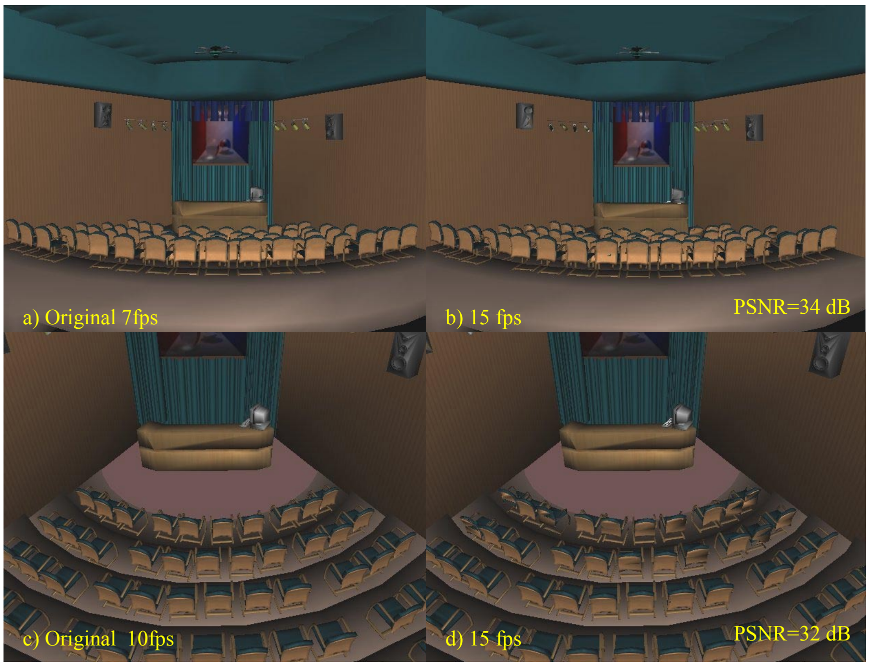

Figure 8.

REFERENCES

[Benic99] F. Benichou, G. Elber, " Output Sensitive Extraction of Silhouettes from Polygonal Geometry,"*Proceedings Seventh Pacific Conference on Computer Graphics and Applications*, pages 60 -69, 1999.

[Brand98] S. Brandt, G. Nutt, T. Berk, and M. Humphrey, "Soft Real time Application Execution with Dynamic Quality of Service Assurance", *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 154-163, May 1998.

[Bril01] R. J. Bril, C. Hentschel, E.F.M. Steffens, M. Gabrani, G. (Sjir) van Loo and J.H.A. Gelissen, "Multimedia QoS in Consumer Terminals," *Invited paper IEEE Workshop on SignalProcessing Systems (SIPS*), Antwerp, Belgium, September 26-28, 2001.

[Cohen98] J. Cohen, M. Olano and D. Manocha, " Appearance-Preserving Simplification," *Proceedings of SIGGRAPH '98*, pages 115-122, 1998.

[Corme90] T.H. Cormen, C. E. Leiserson and R. L. Rivet, " Introduction to Algorithms", *MIT Press, McGraw-Hill*, 1990.

[Funkh93] T. A. Funkhouser and C. H. Sequin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments", *Computer Graphics Annual Conference Series*, pages 247--254, August 1993.

[Garey79] M. R. Garey and D.S. Johnson,"Computers and Intractability: A Guide to the Theory of NP-Completeness", *W.H. Freeman and Company*, New York, 1979.

[Garla97] M. Garland and P.S. Heckbert, " Surface Simplification using Quadric Error Metrics", *Proceedings of SIGGRAPH 97. In Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, pages 209—216, 1997.

[Gobbe99] E. Gobbetti and E. Bouvier, "Time-Critical Multiresolution Scene Rendering", *Proceedings IEEE Visualization 1999*, IEEE Visualization Conference 1999.

[Hafid98] Hafid, G.Bochmann and R. Dessouli, " QoS and Distributed Multimedia Applications: A Review", *The Electronic Journal on Networks and Distributed Processing*, issue 6, February 1998.

[Hoppe97] H. Hoppe, " View-Dependent Refinement of Progressive Meshes," *Proceedings of SIGGRAPH '97*, pages 189-198, 1997.

[Hoppe98] H. Hoppe, " Efficient Implementation of Progressive Meshes," *Technical Report*

MSR-TR-98-02, Microsoft Research, January 1998.

[Lafru98] G. Lafruit, T. Gijbels, A. Scherpenberg, T. Huybrechtsa and J. Bormans, " Complexity analysis of OpenGL 3D rendering for Computational Graceful Degradation," *ISO/IECJTC1/SC29/WG11/MPEG98/M4076*, Atlantic City, October 1998.

[Lafru00] G. Lafruit, L.Nachtergaele, K. Denolf and J. Bormans, "3D Computational Graceful Degradation", *ISCAS 2000 ISCAS - Workshop and Exhibition on MPEG-4, Proceedings*, pages III-547 - III-550, May 28-31, 2000.

[Lee99] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A Scalable Solution to the Multi-resource QoS Problem", *Proceedings of IEEE RTSS'99*, December 1999.

[Luebk97] D. Luebke, C. Erikson, " View-dependent Simplification of Arbitrary Polygonal Environments," *Proceedings of SIGGRAPH '9*7, pages 199-208, 1997.

[Macie95] P. Maciel and P. Shirley, "Visual Navigation of Large Environments Using Textured Clusters", *In Proc. 1995 Symp. Interactive 3D Graphics*, pages 95--102, 1995.

[Pham00] N. Phamngoc, S. Himpe and R. Lauwereins, "An Overview of QoS at End-Systems: Problems, Solutions and Challenges for Future Multimedia Applications", *internal report, ESAT/ACCA*, KULeuven, 2000.

[Sande00] P.V. Sander, X. Gu, S.J. Gortler, H. Hoppe and J. Snyder, " Silhouette Clipping," *Proceedings of SIGGRAPH 200*0, pages 327-334, 2000.

[Schma97a] D. Schmalstieg, "A Survey of Advanced Interactive 3-D Graphics Techniques", Research paper, Institute of Computer Graphics, Vienna University of Technology, 1997.

[Schma97b] D. Schmalstieg and G. Schaufler, " Smooth Levels of Detail", *In Proc. of IEEE 1997 Virtual Reality Annual Intnl. Sym*, pages 12—19, 1997.