CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. Lluch (Editors)

Occlusion ratio: a new query parameter for GPUs

Ignacio Mansa[1], Aiert Amundarain[2], Luis Matey[2], Alejandro García-Alonso[3]
[1] Lander Simulation,[2] CEIT, [3] Euskal Herriko Unibertsitatea
{imansa@landersimulation.com, aamundarain@ceit.es, lmatey@ceit.es, alex.galonso@ehu.es}

Abstract

This work suggests improvements that can be added to current GPUs and graphic rendering APIs to increase occlusion culling performance, that is, to raise the frame rate that can be achieved using hardware occlusion queries. The proposal can be easily implemented in current GPU architectures. It extends the concept of object relevance in an image and introduces the occlusion ratio parameter. Numerical tests have been carried out using pre-processed data to demonstrate our proposal. These tests show that 800% mean frame rate improvements could be achieved if occlusion queries to the GPU would return occlusion ratio data. Moreover, in low occlusion density situations -worst cases with 60-80% objects visible-, results show that occlusion ratio data provides 60-400% improvements in the frame rate.

Categories and Subject Descriptors (according to ACM CCS): I. 3. 1 [Computer Graphics]: Hardware Architecture – Graphics Processors; I. 3. 3 [Computer Graphics]: Picture/Image Generation- Display algorithms; I. 3. 7 [Computer Graphics]: Three-Dimensional Graphics and Realism-Hidden line/surface removal

1. Introduction

For about 25 years most commercial graphic rendering hardware has used depth buffer based techniques to solve the visibility problem. Following this paradigm, current GPUs render at interactive frame rates models described by millions of polygons, for instance, an aircraft engine described by 3.5 millions of polygons is rendered at 15 fps by an nVIDIA 7800 GPU (see Figure 1). However, this powerful image generation capacity is sometimes spent in a non-efficient way. As *direct drawing* (brute force) rendering does not use culling techniques then, many polygons that are drawn in the image buffer do not appear in the final image. Later on the need for culling techniques will be discussed.

Ray tracing algorithms address directly the nucleus of this problem and they provide many achievements [PBMH02, WDS04, WSS05]. However, most applications rely on current GPU architectures, so, in most cases, rendering frame rate is speeded up using culling techniques: frustum, back-face or occlusion culling.

Some applications can also improve frame rate using Level of Detail (LOD) algorithms [LRC*02]. LOD makes use of the concept of *screen relevance*. This property measures the relevance of an object when rendered in a given image. It is usually measured making an estimation of the number of pixels covered by the projection of the object in the image. This paper does not deal with classical LODs, but it makes use and extends the semantics of the image relevance property.

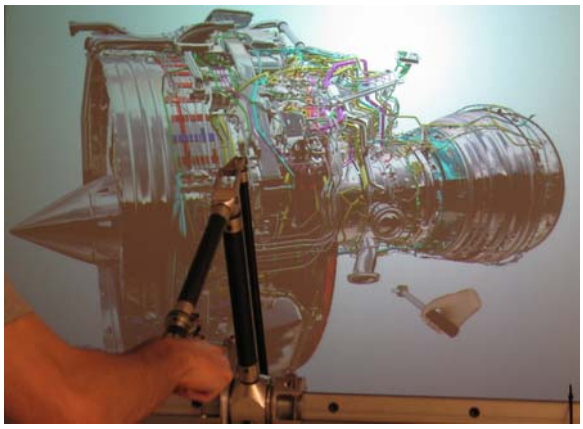

Figure 1: *Digital mock-up of an aeronautical engine.*

Our previous works on occlusion culling were based on "available commercial hardware", and they provided frame rate improvements for the virtual environments we had to deal with. This work suggests a *query extension* that can be easily implemented in commercial GPUs. It also shows the frame rate increments that this extension can provide.

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG09/085-093

The experiments developed to verify the proposal were based on pre-processed data. The rendering process simulates queries to a hypothetical GPU equipped with the extension here proposed. Along the tests, the pre-processed data simulate this GPU.

So, in this research, instead of proposing methods that can be implemented to increase the frame rate, we are proposing an extension to current GPUs that will permit new hardware occlusion query based frame rate improvements. Section 3 presents tests performed using an environment different from our usual mechanical mock-ups.

Before presenting proposals and results, the following paragraphs discuss the need for occlusion techniques.

In *densely occluded* scenes, like urban walkthroughs, up to a 90% of the objects considered for rendering are occluded. Hence, occlusion culling algorithms can identify most of them and the application will only render the remaining 10% of the scene providing an important speed up. There are different types of densely occluded scenes; and different occlusion strategies have been developed for each one of them [TS91, LG95, LSC03].

Our previous research showed that occlusion culling in *not densely occluded* scenes, such as engineering models, often provides an important boost to performances [MAE*06].

Complexity of mechanical models augments steadily. Within the last four years, in our simulations with aircraft engines, models have increased from 0.6 to 3.5 millions of polygons. Not only model complexity has suffered an important increase, but also rendering requirements requested by users (see Figure 2). Simulation of assembly and disassembly tasks for maintainability or training purposes requires a real-time rendering. Even though the speed up is not as great as in an urban walkthrough, the need for interactive simulations encouraged developing simple and fast occlusion culling algorithms.

This visualization module was integrated within a VR system used for maintenance simulation of aircraft engines: man and tool accessibility analysis in order to analyze assembly-disassembly sequences and times (see Figure 1). The virtual environment simulates accessibility in a complex aircraft engine. It uses stereoscopic visualization and a large haptic device [SBGM02, BGM04]. Stereo display is required in order to correctly reach the elements and avoid collisions with mock-up's parts. Using current GPU technology, occlusion culling provides a mean 300% image generation speed up [MAMG08]. As the paper shows, GPUs with the extension here suggested could boost that figure to a 800% (average value), and what is even more important, substantial gains are achieved in worst case situations.

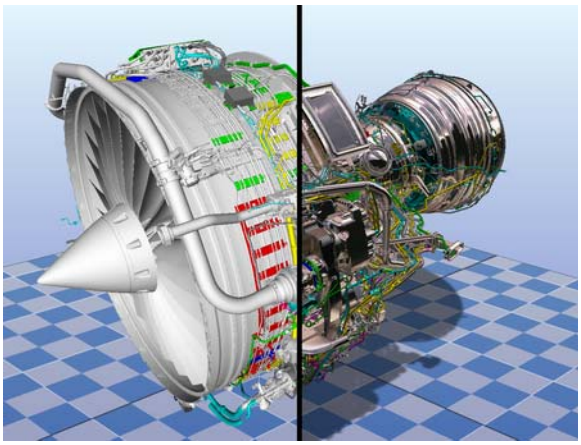

Figure 2: *Plain vs presently expected rendering quality.*

If model complexity and rendering requirements keep the current trend, the performance increments that can be achieved with the extension here proposed will be quite useful. Moreover, current tendency to use GPUs as general purpose co-processors [OLG*05] make this speed ups even more necessary.

Section 2 focuses on the theoretical aspects while section 3 analyses the results achieved.

2. New extension for GPUs

Surveys written by Cohen-Or et al [CCSD03] and Bittner [BW03] provide several techniques to compute the visibility of geometric-sets in virtual environments. Usually occlusion algorithms deal with specific scenarios. Aila and Miettinen [AM04] have implemented an occlusion system that applies different strategies to different scenes.

The proposals discussed in this paper emerged while working in the specific occlusion problem described in the introduction, and the initial tests performed to validate their interest were performed using that visualization environment. The experiments returned such good results that a completely different data set was prepared. It meant moving from the mechanical area towards a natural model (see section 3). Although the visualization system used to test the new environment was the same one used for the mechanical mock-ups, the results were even better.

We expect that there are other virtual environments and their corresponding visualization systems that could also benefit from the proposal.

This section describes the extension suggested for the GPU and how it can be used by any visualization system that makes use of occlusion culling. As there is not any GPU that can be used to assess the extension, a framework had to be built. The extension was added to a "basic"

occlusion query stereo visualization system. The test framework used to obtain the results is explained in section 3 (results). The basic system is not described in detail, because what this paper shows are the gains that can be achieved with the extension. The basic system selected uses the techniques described in section 2.2. It has been extensively tested using many variants and different parameters assessing both frame rate speeds up and image quality [MAE*06, MAMG08]. After all those trials it seemed that the basic system cannot provide larger speed ups, for this reason it was decided to investigate whether new information gathered from the GPU could increase performance.

2.1 Background considerations

Urban and architectural scenes were the first ones to be analyzed [TS91]. Wonka et al. [WWZ*06] have recently published very interesting advances for this type of scenarios.

Some lines of research determine the visibility of the objects in the image space. Zhang et al. [ZMHH97] computed the visibility of an object through two tests: an overlap test and a depth test. Several image space algorithms have been included by the manufacturers in the graphic systems to accelerate visibility computations. ATI presented the HyperZ technology [Mor00], where the depth operations were optimized. Another feature implemented was the HP occlusion flag, which allows feedback of occlusion information for drawing objects. Klosowski and Silva [KS01] developed an occlusion algorithm employing this feature.

Other authors have proposed extensions to GPUs. For instance, Bartz et al. [BMH99] suggested the occlusion query that current GPUs integrate. Similarly Aila et al. [AMN03] proposed the delay streams architecture that improves occlusion computing performance.

The last generations of graphic hardware are supplied with a new feature: the nVIDIA occlusion query. Besides determining the visibility of a geometry set, it also informs about how many pixels are visible. In addition it offers mechanisms for using it without breaking the synchronism between the CPU and the GPU. A bad scheduling in the use of the occlusion queries can break that synchronization and that will entail a slowdown in the performances [BWPP04]. Three GPUs solve this problem using the "occlusion-switch" proposed by Govindaraju et al. [GSYM03]. Mattausch et al. [MBW08] addresses also this problem by using adaptive visibility prediction and query batching.

2.2 Basic system

This subsection does not consider the extension; it resumes the main characteristics of an occlusion system which is described with more detail by Mansa et al. [MAE*06, MAMG08]. This system will be called *basic system* because the extension was built on top of this software.

Occlusion culling methods based on current GPU features take advantage of different properties that can be found in the virtual environments or in the way they are visualized.

The basic system uses temporal coherence (frame to frame) to achieve an efficient occlusion query scheduling, similar to that used by other authors [BWPP04, Sek04]. As visibility information between two consecutive frames does not vary too much; the information generated in the previous frame can be used as an efficient start-point in the following one.

The system uses temporal coherence independently for each eye instead of stereo coherence. This strategy might be specific to certain virtual environments, because other authors use stereo coherence in their occlusion culling strategies [WZQK04]. Anyway, whether stereo coherence is used or not, it should not affect the improvements provided by the extension.

The system keeps a quasi-depth sorted list of the visible objects using a simple strategy. The sorting criterion is based on the distance between an object's AABB and the viewpoint. So, objects that are good occluders are drawn before testing occlusion for those objects that are probably occluded.

In fact, as the viewpoint moves, most frames do not need a depth sort. A new sort operation is performed only when a motion threshold is surpassed. The algorithm is inspired in the temporal coherence strategy implemented to improve view frustum culling by Güdükbay et al. [GY02].

When using hardware occlusion query bounding boxes are often used as test geometry to query if the object is occluded [KS01, Stan05, YSM03]. Bartz et al. used k-dops and convex hulls [BSK05]. The last version of DirectX (v10), allows users to define the preferred geometry to perform the test. Previous to this release, the basic system used a simplified version of the objects as the geometry for the occlusion tests. The occlusion query efficiency trades between drawing a few triangles, that generate an oversized area projection; and drawing about a thousand polygons, whose projection generates a better fitted area. Results showed clearly that using simplified geometry is more efficient. This is due to the shape of the objects such as pipes and other twisted elements. Simplified models are generated using Garland simplification method [GH97], but other techniques could also be integrated [BNR*01, BKE00]. A 75% simplification level is used. Higher

percentages of simplification compromise the quality of the model generated: the deformation in the shape of the models could invalidate the results of the occlusion queries. In addition, these simplified models are also used when the objects are barely visible in the final image.

Virtual environments that are depth enough make use of level of detail (LOD) techniques: the relevance that the projection of one object has in the screen (screen relevance). Compact scenarios are not depth enough to take advantage of these algorithms. However, the basic system extends the semantics of the LOD paradigm: the relevance that one object has in the image (image relevance). The basic system draws a simplified model of the object when the occlusion query reports that the object is visible in less than 125 pixels (0.01% of a 1280x1024 drawing area). In this case it is assumed that the object is very small or that most of the object is occluded. As in either case the object has little relevance for the final image, it is drawn using a simplified version of the object.

The occlusion query scheduling and the space sorting mechanism are conservative methods; they do not introduce any error on the final image (compared with the direct drawing alternative). On the contrary, the test geometry and LOD based image relevance may introduce some errors. Errors were measured numerically and with user trials. Both techniques reported that image errors are negligible [MAE*06].

However, sometimes these techniques do not provide enough frame rate speed up. Analysis show that for a given mock-up occlusion density varies between a 20% and an 80% depending on the position and orientation of the point of view. Low occlusion situations notably reduce the visualization frame rate. An approach to solve this problem has been aggressive rendering. This makes it possible to trade between frame rate improvements and possible small visual artifacts.

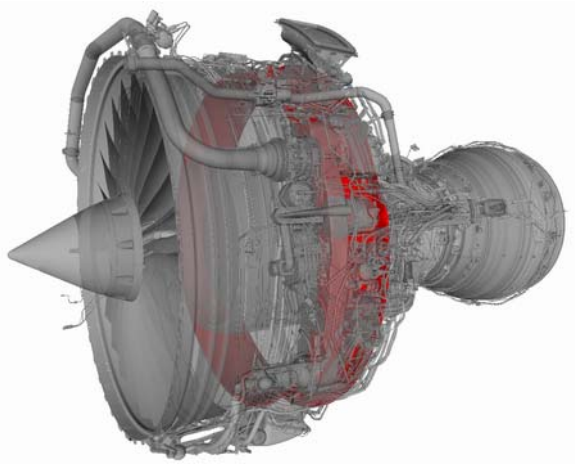

Figure 3: *Low image relevance object.*

The probability that an occluded object is also occluded during the following frames is high. Aggressive rendering is performed in each new frame by considering as occluded, objects that were occluded in the previous frame. Thus, only a fraction of the occluded objects in one frame are tested in the following frame to find if they are occluded or not. This makes it possible to trade between frame rate improvements and possible small visual artifacts. These flaws are due to objects that become visible and whose occlusion status is not checked for a number of frames. This number of frames is the function of the fraction of occluded objects tested in each frame: set to 50% means that a new visible object may not appear in one frame, 25% can produce three frame faults, and so on. The analyses in this paper were done using a 50% (aggressive rendering) and a 100% (no aggressive rendering). Tests using other fractions are reported elsewhere [MAMG08], for instance a 25% fraction did not provide substantial frame rate improvements and 10% often created noticeable artifacts.

Pursuing even greater frame rate improvements, new aggressive methods were considered, but they require the extension that is described in the following paragraphs.

2.3 Occlusion query extension: definition and use

Searching for aggressive occlusion techniques, we realized a recurrent fact: some objects that cover many visible pixels in the final image have little relevance in this final image. This feature is depicted by the red object in Figure 3. The figure renders the object itself covered by a transparent image of the whole model. It gives an idea of the “relevance” of the object in the image and it also gives an idea of the high number of pixels actually covered by the object in the final image. This situation happens not only to this large object, but to many other.

In compact scenes, the objects are not far away from the viewpoint; but there is a hierarchy of “image interest” (relevance). The objects that are partially hidden generally have less interest for the user. Although these objects cover a significant region in the image, the simple fact of being behind many other objects reduces their relevance. In these cases, the number of visible pixels does not reflect “the real” contribution of the object in the image, since this number of pixels does not manifest the inter-relation among objects.

The figure teaches us that knowing the absolute number of visible pixels of one object in an image is not always enough to know how high is its relevance in the image. In order to have more data to improve the evaluation of image relevance, it is needed to know a relative value, that is, the number of pixels that are visible compared with the total number of pixels that the projection of an object covers in the image.

Currently, GPUs return the number of visible pixels in answer to an occlusion query. However, placing that query does not provide the number of pixels covered by the projection of the object (visible plus not visible). It seems that counting the number of covered pixels does not add a heavy burden to the GPU. Computing this value should not penalize the process of evaluating the number of visible pixels.

The GPU could count the number of projected pixels at the same time that counts which of those projected pixels are visible. Then, occlusion queries could return both figures. Finally, the application would be able to use them to evaluate image relevance and make decisions about how to render the object.

It is clear that, knowing these two values, new techniques can safely make wise decisions, which without compromising image quality, will provide frame rate speed ups.

A new scheme can be implemented based on the following parameter that makes use of the proposed extension:

$$Occlusion\_Ratio = \#visible \ / \ \#projected \quad (1)$$

The intended idea is to draw with low detail objects that have a small quotient, that is, to draw with low detail objects that have a small visible region in comparison with their size. This can be safely done because the visual interest of partially occluded objects is low.

However, if only a small portion of the object is visible (its ratio is less than, for example, 30%) but its projection occupies a big portion of the image, it does not seem reasonable to draw it with a simplified version.

So, two restrictions must be considered when using the occlusion ratio parameter to draw an object with a simplified model: Occlusion_Ratio must be lower than the selected threshold, and the amount of visible pixels of the object must be less than, for instance, 0.15% of the drawing area. Following the initial trials a 0.3 threshold was selected for the validation experiments

Next section presents results from experiments that verify the scheme here proposed. The new algorithm, Occlusion Ratio Algorithm (ORA), has been developed using the "basic system" and applying the pseudo-code presented in Figure 4.

```
if ( Occlusion_Ratio  <  threshold     &
     visible pixels   <  0.15% drawing area )
then draw simplified object
else draw object normally
```

Figure 4: *ORA scheme*

3. Results

In this section the methodology used to validate our proposal, the new query extension, is explained and the most important results are also discussed.

3.1 Methodology

The procedure followed allows us studying the behavior of the algorithms with different models and camera paths. As the camera moves automatically along the desired path, several data can be measured in a controlled way. In every frame the whole model is located within the viewing frustum, therefore view frustum culling itself does not affect the experiments.

The experiments assess the quality of each individual image and "its frame rate". The frame rate is not an averaged value; instead it is "discretely" measured for each image. Quality is evaluated by measuring the percentage of wrong pixels in the image. This information is very intuitive to be aware of the algorithm performance.

In order to validate the proposition and considering that the occlusion ratio data is not provided by GPUs at the moment, the following experiment has been carried out with several models and paths.

The experiments consider that the number of pixels cast by the object, needed to compute the occlusion ratio quotient, is already available. In a pre-process step, the number of pixels projected by each object is computed and stored for each position along the path. Later, in real time, the quotient is computed with the number of pixels visible provided by the GPU and the number of pixels cast by the object computed in pre-process.

The rendering features simulate realistic lighting inside a workshop with four ambient lights and environment mapping reflections.

The experiments have been carried out using a single PC configured with one Pentium 4 at 3.2 Ghz, 2 Gb of RAM and running under Windows XP. The frames were rendered in stereo at a resolution of 2x1280x1024 on an nVIDIA GeForce 7800 GTX 256 RAM and PCI-Express bus.

Figure 5: *High detailed polygonal Oak Tree model.*

3.2 Frame rate improvements

Two types of environment have been studied with ORA: high detailed polygonal trees and complex mechanical sets. This section presents results from two models, one of each type. Similar results have been achieved with other trees and aircraft engines.

The first example is a single complex tree (see Figure 5). The different components of the tree (trunk, branches and leaves) are represented with detailed polygonal models instead of textured-mapped polygons. All leaves of each branch are grouped into a single object. The Oak Tree is composed of more than 400 objects and 5 M polygons.

Figure 6 presents the frame rate achieved by direct drawing, the basic system and two ORA strategies. As explained in section 2.2, ORA100% is the conservative version, while ORA50% uses a more aggressive culling strategy. The occlusion density for this model along the camera path ranges between a 20% and a 70%. That is, in the worst cases, 80% of the objects are visible.

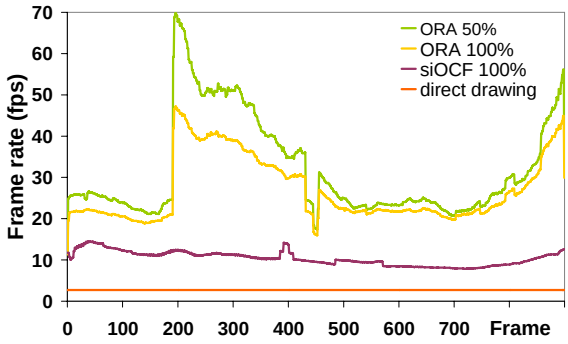

Figure 6: *Frame rate with Oak Tree model.*

The basic system improves the performances (an average 283% of speed up) compared to direct drawing but the frame rate remains modest. On the other hand, frame rate improvements by the ORA approaches are notable and allow interactive simulations with stereoscopic visualization. ORA100% provides a significant 895% average frame rate speed up relative to direct drawing. When analyzing a fraction of 50% (of the occluded objects of the previous frame), ORA50%, the results reach an outstanding 1079% performance boost.

The second example uses an aircraft engine model (see Figure 7). This aircraft turbine is composed of 2300 objects and 3M polygons. Mechanical sets are compact scenes with thousands of elements connected between them. The geometrical complexity can easily reach several millions of polygons. Aircraft engines are enveloped by casings that are usually hidden by the externals of the turbine (pipes, wires, gauges, etc.).

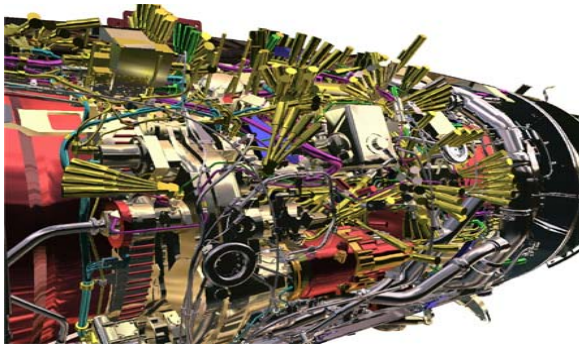

Figure 7: *Complex mechanical set:Turbine model.*

Two camera paths with different occlusion density are here presented to validate our proposal with Turbine model.

The occlusion density for Turbine in the first camera path ranges between 65% and 80%. Figure 8 shows the frame rate achieved with direct drawing, the basic system and two ORA strategies.

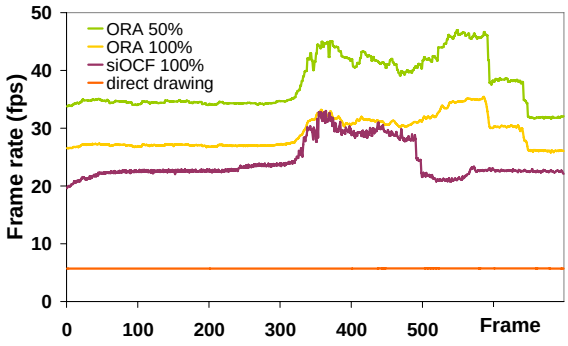

Figure 8: ***Frame rate with Turbine model (path 1).***

The basic scheme provides a 322% average frame rate

speed up compared to direct drawing. When the ORA100% is applied a 410% performance boost is obtained. Results using a more aggressive version of the algorithm, ORA50%, achieve a significant 560% frame rate speed up.

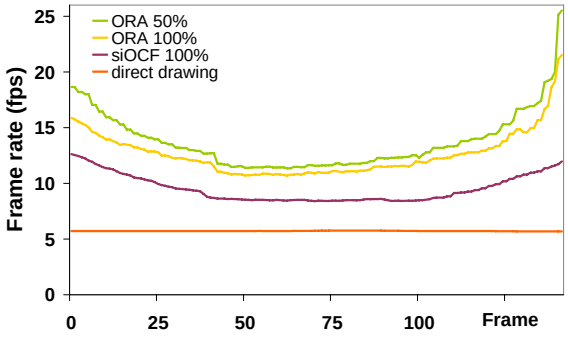

Figure 9: ***Frame rate with Turbine model (path 2).***

The second path focuses on worst case regions of the same model. These regions have low occlusion density, about a 40%, and make difficult interactive rendering. Figure 9 shows the frame rate achieved by the different approaches.

The basic system brings attractive performance with high occlusion density regions (path 1). However in critical regions of Turbine (path 2) it provides much lower speed ups, around a 60%. On the contrary, ORA100% offers a 109% average frame rate speed up compared to a direct drawing. Results using a more aggressive version of the algorithm, ORA50%, achieve an interesting 127% frame rate speed up for these low occlusion viewing situations.

The worst speed up case found in all the experiments with different models and paths was a 60%. It happened with a turbine made up of 2.500 elements and 1.6M polygons, less than half the size of the turbine here presented. In that specific camera position (the worst speed up found) we also encountered one of the lower occlusion density situations.

It can be asserted, that even in critical regions, the use of the occlusion ratio data provides substantial increases in the frame rate such as to make valuable the extension here proposed.

3.3 Image quality

The loss of quality in the images has been quantified. The images generated with a conservative occlusion strategy (fraction of 100%) have one error source when compared to the "exact" images generated without occlusion culling: some objects are drawn using simplified geometric models. On the other hand, using fractions less than 100% of the occluded objects for visibility analysis can produce popping artifacts.

The quality of the images with both models was similar. Table 1 gathers error statistics.

Stats (%)	Basic system	ORA100%	ORA50%
Mean	0,002	0,002	0,005
Min	0	0	0
Max	0,023	0,036	0,531
StdDev	0,004	0,006	0,027

Table 1: *Analysis of wrong pixels generated in the image.*

The basic system generates an insignificant percentage of wrong pixels. ORA strategies degraded image quality a little bit more. With a fraction of 50%, the average error is doubled. The maximum error measured is about 0.5% but the standard deviation confirms that these differences occur only in few frames. The percentage of wrong pixels seems negligible. Even more, users, while involved in their simulation tasks, do not notice these small artifacts. Following these objective and subjective facts, we can conclude that the use of occlusion ratio data provides a good image quality.

4. Conclusions and future work

A new parameter for hardware occlusion queries has been proposed and validated. Using the occlusion ratio parameter, applications can discriminate objects that are visible in a high number of pixels, but whose relevance in the image is not high.

It seems to be easy to implement the proposed new parameter in GPUs and in graphic APIs.

The method described uses an aggressive strategy without compromising image quality. Tests show a mean 800% frame rate speed up, including worst cases.

Worst case situations are caused by low occlusion density. The experiments have encountered situations where an 80% of the objects were visible. Frame rate speed ups are even more necessary in these cases, and the occlusion ratio parameter has shown its usefulness.

These results manifest the benefits that could be achieved if the occlusion ratio query were implemented in the GPUs.

The occlusion ratio parameter can be used with other environments different from those analyzed here, provided that a convenient number of objects are often partially occluded by others. The query can be easily exploited by scene graphs and other visualization algorithms.

Acknowledgements

The research work presented in this paper is supported by the European Commission, under the FP6 IST-2002-002114 Enactive NoE (http://www.enactivenetwork.org). Dr. Garcia-Alonso research was supported by the Spanish Ministry of Educational and Science, grant TIN2006-14968-C02-01.

References

[AM04] Aila T., Miettinen V.: dPVS: An occlusion culling system for massive dynamic environments. Comp. Graph. & App., 24,2 (2004): 86-97.

[AMN03]Aila T., Miettinen V., Nordlund P.: Delay streams for graphics hardware. ACM Transactions on Graphics, 22,3(2003): 792-800.

[BMH99]Bartz D., Meißner M., Huttner T.: Opengl-assisted occlusion culling for large polygonal models. Comp. and Graph., 23,5(1999): 667-679.

[BSK05] Bartz D., Staneker D., Klosowski J.: Tighter bounding volumes for better occlusion performance. Tech. report, U. of Tübingen. 2005.

[BKE00] Bernardini F., Klosowski J.T., El-Sana J.: Directional discretized occluders for accelerated occlusion culling. Computer Graphics Forum, 19,3 (2000): 507-516.

[BWPP04] Bittner J., Wimmer M., Piringer H., Purgathofer W.: Coherent hierarchical culling: Hardware occlusion queries made useful. Computer Graphics Forum, 23,3(2004): 615-624.

[BW03] Bittner J., Wonka P.: Visibility in computer graphics. Environment and Planning B: Planning and Design, 30,5(2003): 729-756.

[BGM04]Borro D., García-Alonso A., Matey L.: Approximation of optimal voxel size for collision detection in maintainability simulations within massive virtual environments. Computer Graphics Forum, 23,1(2004): 13-23.

[BNR*01] Brunet P., Navazo I., Rossignac J., Saona-Vázquez C.: Hoops: 3d curves as conservative occluders for cell-visibility. Computer Graphics Forum, 20,3(2001).

[CCSD03] Cohen-Or D., Chrysanthou Y.L., Silva C.T., Durand F.: A survey of visibility for walkthrough applications. Visualization and Computer, IEEE Transactions on Graphics, 9,3 (2003): 412-431.

[GH97] Garland M., Heckbert P.S.: Surface simplify-cation using quadric error metrics. Comp. Graph. (Proc. SIGGRAPH'97), 1997, 209-216.

[GSYM03] Govindaraju N.K., Sud A., Yoon S.E., Manocha D.: Interactive visibility culling in complex environments using occlusion-switches. *Symp. on Interactive 3D graphics*,103-112, 2003.

[GY02] Güdükbay U., Yilmaz T.: Stereoscopic view-dependent visualization of terrain height fields. IEEE Transactions on Visualization and Computer Graphics, 8,4(2002): 330-345.

[KS01] Klosowski J., Silva C.T.: Efficient conservative visibility culling using the prioritized-layered projection algorithm. IEEE Trans. on Visualization and Comp. Graphics, 7,4 (2001): 365-379.

[LSC03] Leyvand T., Sorkine O., Cohen-Or D.: Ray space factorization for from-region visibility. ACM Transactions on Graphics, 22,3(2003): 595-604.

[LG95] Luebke D., Georges C.: Portals and mirrors: Simple, fast evaluation of potentially visible sets. In *Proc Symposium on Interactive 3D Graphics*, pp. 105-106, 1995.

[[LRC*02] Luebke D., Reddy M., Cohen J., Varshney A., Watson B., Huebner R.: *Level of detail for 3d graphics*. Morgan Kaufman, 2002.

[MAE*06] Mansa I., Amundarain A., Elizalde E., Garcia-Alonso A., Matey L.: Towards adaptive occlusion culling using camera coherence. *Proc Information Visualization*, pp. 591-596, 2006.

[MAMG08] Mansa I., Amundarain A., Matey L., García-Alonso A.: Analysis of coherence strategies for stereo occlusion culling. Computer Animation and Virtual Worlds, 19,1 (2008):67-77.

[MBW08] Mattausch O., Bittner J., Wimmer J.: CHC++: Coherent hierarchical culling revisited. Computer Graphics Forum, 27,2(2008): 221-230.

[Mor00] Morein S.: Ati radeon hyper-z technology. *Proc Hot3D- Graphics Hardware Workshop*, 2000.

[OLG*05] Owens J.D., Luebke D., Govindaraju N.K., Harris M., Krüger J., Lefohn A.E., Purcell T.J.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum, 24,3(2005): 21-51.

[PBMH02] Purcell T.J., Buck I., Mark W.R., Hanrahan P.: Ray tracing on programmable graphics hardware. In *Proc 29 th Computer graphics and interactive techniques*, pp. 703-712, 2002.

[SBGM02] Savall J., Borro D., Gil J.J., Matey L.: Description of a haptic system for virtual maintainability in aeronautics. In *Proc International Conference on Intelligent Robots and Systems (IROS)*, pp. 2887-2892, 2002.

[Sek04] Sekulic D.: *Efficient occlusion culling* in *Gpu gems*, Chapter 29. R. Fernando. 2004.

[Stan05] Staneker D.: Hardware-assisted occlusion culling for scene graph systems. Phd thesis., Eberhard-Karls-Universität Tübingen2005.

[TS91] Teller S.J., Sequin C.H.: Visibility preprocessing for interactive walkthroughs. Computer Graphics (Proc. of SIGGRAPH '91), 25,4(1991): 61-69.

[WDS04] Wald I., Dietrich A., Slusallek P.: An interactive out-of-core rendering framework for visualizing massively complex models. *Proc Eurographics Symposium on Rendering*, pp. 81-92, 2004.

[WZQK04] Wang M., Zhang N., Qu H., Kaufman A.E.: Interactive stereoscopic rendering of volumetric environments. IEEE Trans. on Visualization and Computer Graphics, 10,1(2004): 15-27.

[WWZ*06] Wonka P., Wimmer M., Zhou K., Maierhofer S., Hesina G., Reshetov A.: Guided visibility sampling. ACM Trans Graph, 25,3(2006) 494-502.

[WSS05] Woop S., Schmittler J., Slusallek P.: Rpu: A programmable ray processing unit for realtime ray tracing. In *Proc Computer Graphics and Interactive Techniques-ACM SIGGRAPH 2005 Papers*, pp. 434-444, 2005.

[YSM03] Yoon S.-E., Salomon B., Manocha D.: Interactive view-dependent rendering with conservative occlusion culling in complex environments. *IEEE Visualization Conference*, 2003.

[ZMHH97] Zhang H., Manocha D., Hudson T., Hoff III K.E.: Visibility culling using hierarchical occlusion maps. Computer Graphics (Proc. of SIGGRAPH '97), 1997): 77-88.