Efficient Occlusion Culling using Solid Occluders

Georgios Papaioannou Athanasios Gaitatzes Dimitrios Christopoulos

Foundation of the Hellenic World
Poulopoulou 38
11851, Athens, Greece

gepap@fhw.gr gaitat@fhw.gr christop@fhw.gr

ABSTRACT

Occlusion culling is a genre of algorithms for rapidly eliminating portions of three-dimensional geometry hidden behind other, visible objects prior to passing them to the rendering pipeline. In this paper, an extension to the popular shadow frustum culling algorithm is presented, which takes into account the fact that many planar occluders can be grouped into compound convex solids, which in turn can provide fewer and larger culling frusta and therefore more efficient elimination of hidden geometry. The proposed method combines planar and solid occluders using a unified selection approach and is ideal for dynamic environments, as it doesn't depend on pre-calculated visibility data. The solid occluders culling algorithm has been applied to commercially deployed virtual reality systems and test cases and results are provided from actual virtual reality shows.

Keywords

Hidden surface removal, visibility, virtual reality, games, dynamic environments.

1. INTRODUCTION

In dense or large three-dimensional environments, the visible geometry consists of many hundreds of thousands of polygons. Fortunately, most of these primitives are not visible simultaneously from an arbitrary vantage point. If the amount of actually invisible (hidden) polygons that are nevertheless sent to the graphics hardware for rendering is kept low, the performance of the application can gain a significant boost. The algorithms used for culling these surfaces fall into three categories, back face elimination, view frustum culling and occlusion culling, the later being not so trivial as the rest.

Occlusion Culling

In occlusion culling, geometry that is hidden behind objects closer to the camera point is discarded before being subject to depth sorting algorithms, although this geometry may pass the other two culling tests.

Conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

Most occlusion culling methods use planar occluders [Hud97a] and their projections on the viewing plane. These planar occluders are often the polygons of the blocking geometry. Culling tests take place either in 3D space (viewer coordinate system, see Fig. 1) or in the resulting discrete image space, after projecting and rasterising the occluders and the potentially hidden geometry in an image buffer. The Hierarchical Z-Buffer [Gre93a] and the Hierarchical Occlusion Map [Zha98a] are two good examples of the later approach. Other techniques rely on the concept of space partitioning in cells, which are visible through openings, named portals, as in [Lue95a]. Binary Space Subdivision is often utilized to automatically segment space into cells based on the geometry [Tel91a]. The related visibility culling techniques may use pre-calculated information or clip entire cells on the fly. BSP and cell-and-portal techniques work very well for enclosed environments, such as building interiors and are thus favored by the game development community. Occlusion culling techniques may pose some restrictions on navigation freedom (non-arbitrary view location) in order to gain simplicity and performance, as is the case in [Dow01a]. A more extensive presentation of the different occlusion culling methods is clearly beyond the scope of this paper but the interested reader can find a thorough and comprehensive survey and comparison of occlusion culling algorithms in [Coh03a].

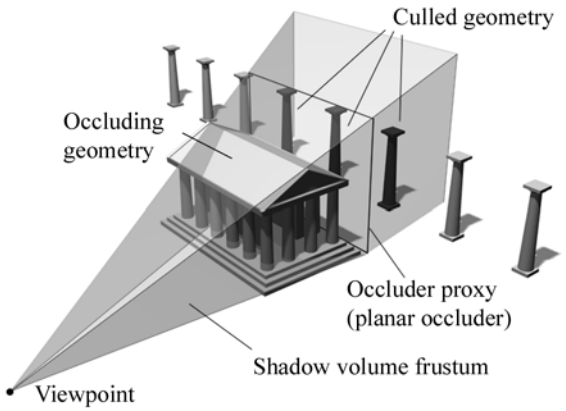

Figure 1. Principle of the shadow frustum culling method.

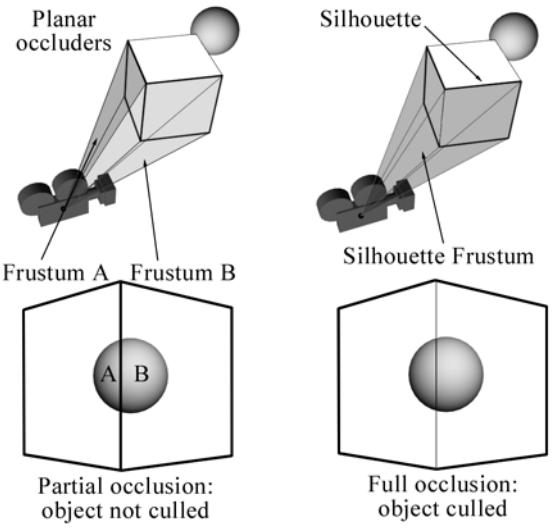

Figure 2. The generation of the culling frusta from convex solid silhouettes eliminates many cases of partial occlusion.

Occlusion tests using small, independent occluders, such as geometry polygons or view-dependent planar occluders, are inefficient because an object may be partially hidden by two or more occluders and therefore not discarded, but totally hidden by the joint surface that the occluders may produce (Fig. 2). To address this, fusion algorithms have been proposed by many authors, as in [Sch00a] and [Kol00a]. These methods try to combine the resulting frusta (in 3D space) or produce aggregate footprints (in image space) of the occluders in order to maximize the culling effectiveness of closely packed but otherwise unrelated occluders. Joining either the view-dependent occluders or the resulting frusta or projected polygons (image-space footprints) involves expensive operations that have to be performed at each frame. Furthermore, containment tests with non-convex (or multiple) frusta or non-convex polygons that are generated from the joint projected occluders are inherently more complex and time consuming when compared to simple individual convex occluder tests. Joining and visibility testing can be simplified by generating approximate convex aggregations of occluder clusters but in this case, an error is introduced which may become intolerable, especially when the viewer comes close to the occluders.

The method for occlusion culling presented here relies on the use of convex solid occluder shapes in a scene to block away geometry fragments (or hierarchies of objects) from arbitrary view points. It extends the work of Hudson et al [Hud97a] for occlusion culling using shadow frusta to address the inefficiency of partial visibility discussed above, while avoiding the expensive joining calculations. Shadow frustum culling uses – usually convex - planar occluder polygons, which can be part of the visible scene geometry or dummy (proxy) occluders, in order to create semi-infinite conical frusta with edges emanating from the viewpoint and passing through the polygon's vertices (Fig. 1). A geometrical entity, which can be anything from a simple polygon to a scene-graph hierarchy of complex objects, is blocked if its bounding volume is completely contained within the frustum. In the case of a hierarchy of objects, if it cannot be safely discarded as a whole due to partial containment into the shadow frustum, the tree is traversed in search of a potentially fully hidden sub-tree. Hudson et. al. [Hud97a] proposed a dynamic occluder selection mechanism which maintains a small list of most efficient occluders for each frame, selected among the full set of planar occluders using proximity, alignment and area coverage criteria. More about this selection mechanism and its extension to the solid occluder paradigm will be discussed in section 3.

Motivation

In many static or dynamic environments and especially in the case of outdoor scenes, where the geometry is often blocky (buildings, cars etc), clustered and possibly sparse, simple occlusion culling performs very poorly due to the partial occlusion phenomenon (Fig. 2): Two or more adjoining planar occluders may partially hide a distant object but the combined occlusion area of them may hide it completely. This object cannot be eliminated if the frustum for each occluder is created separately, and the joining of frusta is, as mentioned, an expensive operation. Pre-calculation methods solve this problem but cannot be effectively applied to dynamic environments.

2. METHOD OVERVIEW

To overcome this limitation, instead of individual planar polygons, convex solid occluders were used, such as boxes and cylinders as visibility blocking proxies for large isolated structures. For the moment, solid occluders are manually defined (modeled). A convex solid, when projected on an arbitrary plane, is guaranteed to produce a convex polygon. The convex frustum of the projected polygon is the union of the frusta that would be generated from the individual planes of the faces of the solid occluder, thus bypassing the need to merge frusta in order to avoid partial occlusion.

Solid occluders are easy and efficient to define in open space environments and therefore the method is most suitable for such scenes. In enclosed (indoor) static environments, although the application of the method is equally feasible, it does not provide any gain when compared with the simpler cell-and-portal occlusion culling schema. Shadow frustum culling using solid occluders is inherently compatible with moving geometry as the solid blockers can be transformed in space in the same manner as the rest of the geometry, without any computational overhead.

For each frame, the solid occluder frustum is generated as follows (Fig. 3). The view-dependent silhouette of the solid occluder is extracted by determining the edges belonging to adjacent polygons, which are not both visible or hidden simultaneously:

$$edge(tr_i, tr_j) \in Silhouette \Leftrightarrow \left[\vec{N}_i \cdot (\bar{P}_{i0} - \bar{C})\right] \cdot \left[\vec{N}_j \cdot (\bar{P}_{j0} - \bar{C})\right] \leq 0 \quad (1)$$

where $\vec{N}_k$ is the normal vector of triangle tr_k, $\bar{P}_{k0}$ the triangle's first point and $\bar{C}$ the viewpoint.

The silhouette edges do not lie on the same plane in general. Therefore, a cap (near plane) for the semi-infinite frustum must be approximately constructed, based on the relative position of the viewer and the silhouette points. If the near plane is too close, geometry that is contained in the convex hull of the edge poly-line may be falsely eliminated. Similarly, choosing a plane near the average of the silhouette points can be a disastrous selection for elongated solid occluders in the view direction for the same reason (Fig. 4). Therefore, the near plane of the frustum is chosen so that it passes through the furthest point of the silhouette line relative to the viewer. The normal vector of the near frustum cap is the average directional vector between each silhouette point and the viewpoint $\bar{C}$:

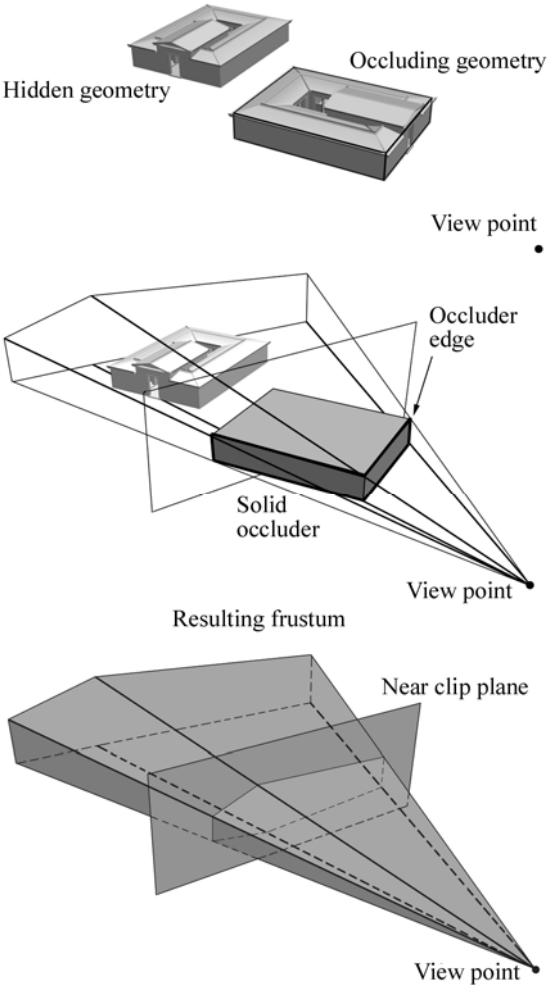

Figure 3. Frustum creation, from silhouette determination to (convex) frustum generation. The near cap of the frustum passes from the furthest silhouette point to the camera position.

$$\vec{N}_{near} = \sum_{i=0}^{K-1} \left(\bar{C} - \bar{S}_i\right) \Big/ \left\| \sum_{i=0}^{K-1} \left(\bar{C} - \bar{S}_i\right) \right\| \quad (2)$$

where $\bar{S}_i$ is the *i*-th point out of the *K* silhouette vertices.

The effectiveness of the solid occluders becomes apparent when the viewpoint is moving among buildings or other isolated obstacles at inspection distance (near). Planar occluders would mostly produce partial occlusion when the viewer is not facing the main sides of the geometry straight on. Most of the time, we view structures and moving objects from odd angles and this is where the solid occluders technique provides a unified and contiguous frustum to take into account all planar sides at once.

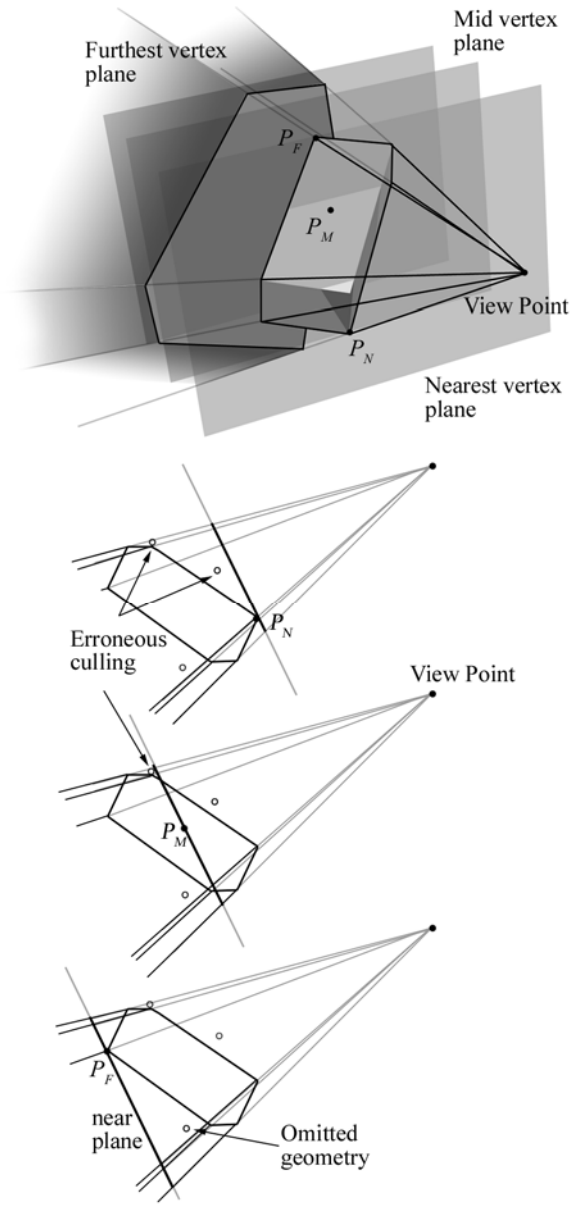

Figure 4. Near frustum cap selection.

3. OCCLUDER SELECTION

As explained by Hudson et al. [Hud97a], a scene may contain too many occluders for a real-time application to be able to test each object against each one of them. For instance, in a finished virtual reality production, such as the "Walk through Ancient Olympia" by the Foundation of the Hellenic World (FHW) [Gai04a], a virtual world may contain more than 200 occluder planes and solids. Therefore, an optimal set of occluders has to be selected for each frame at run-time in order to keep the number of "active" occluder primitives to a minimum. For this task, a ranking function f_{planar} and f_{solid} has to be devised for each type of blocking primitives that takes into account the solid angle of the frustum. For planar occluders, Hudson et al. use the area-angle approximation presented by Coorg and Teller [Coo97a]:

$$f_{planar} = \frac{-A\vec{N}\cdot\vec{V}}{\left\|\vec{V}\right\|^2} \qquad (3)$$

where A is the area of a planar occluder, $\vec{N}$ is its normal vector and $\vec{V}$ is the vector from the viewpoint to the center of the occluder. The criterion of eq. (3) provides a reliable measure of occluder importance. For solid occluders we use an approximation formula, which depends on the projection on the view plane of the solid occluder's volume *Vol* and the squared distance of the occluder from the viewpoint:

$$f_{solid} = \frac{Vol}{\left\|\vec{V}\right\|}\cdot\left\|\vec{V}\right\|^{-2} = \frac{Vol}{\left\|\vec{V}\right\|^3} \qquad (4)$$

Note that the above ranking function for the solid occluders does not depend on angular attributes as the near plane of the constructed frustum always faces the center of projection (see eq. (2)). f_{planar} and f_{solid} are balanced and do not need further biasing to become compatible. At each frame cycle, an active set of occluders, consisting of both solid and planar ones, is determined by evaluating and sorting the result of the corresponding ranking function for each one of them. Please note that due to time coherence and the fact that the occluder selection is not critical for the visual quality, it is not necessary to perform this sorting task at every frame. A typical size for the active occluders set is 10 – 20 occluders, depending on the field of view. For instance, our framework, running on a four-wall surround-screen platform, uses a set of 15 active occluders.

4. CASE STUDIES AND RESULTS

Various tests were run to compare the performance of the solid occluder algorithm with the one of the planar occluder scheme. As both planar and solid occluders are supported and incorporated in our application software that is used to drive the virtual reality shows at FHW, we were able to isolate the two cases under exactly the same application execution scenarios. The integration of the occlusion techniques within a complex scene-graph platform (SGI OpenGL Performer-based) also allowed for realistic performance testing, taking advantage of all the other optimizations available, such as frustum culling and location-sensitive geometry switching. The testing and benchmarking themselves took place

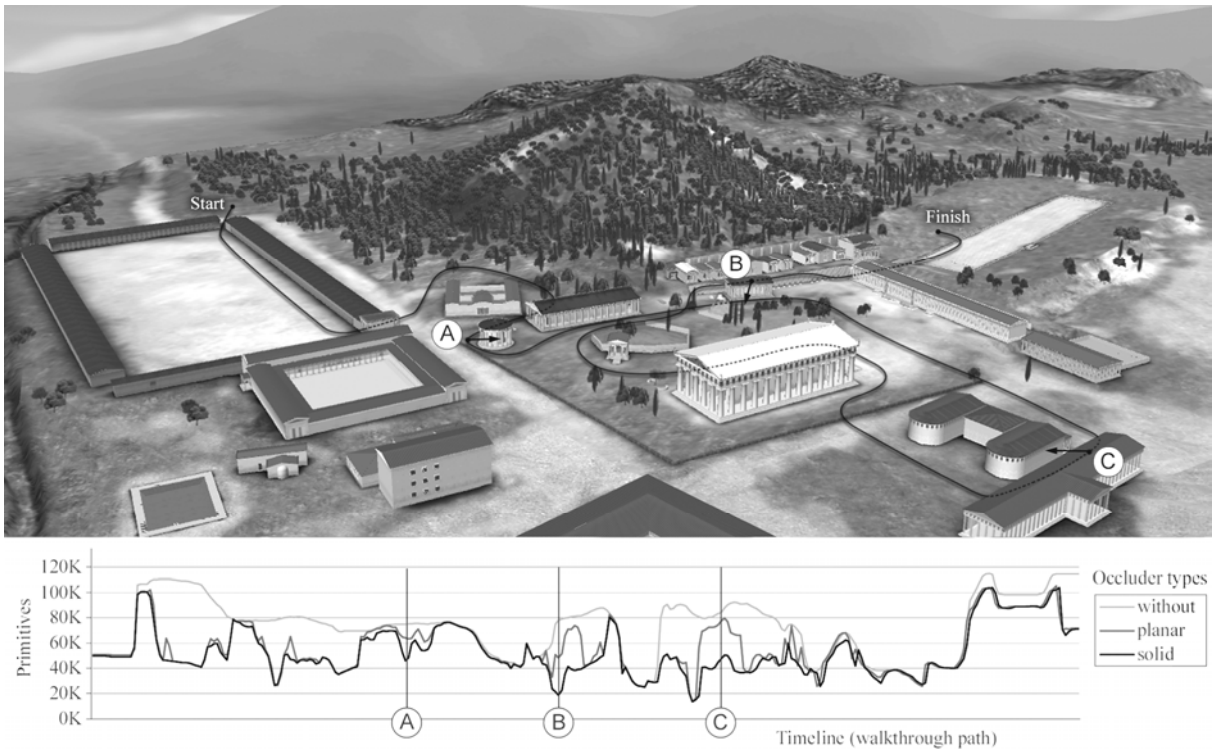

Figure 5. Sparse environment test case – Ancient Olympia. Solid occluders offer a significant culling performance improvement when the camera is moving near structures or when inspecting objects.

in a four-wall CAVE-like surround screen virtual environment. This means that view-frustum culling alone was not a very effective acceleration technique as the largest portion of the world was visible at any time in at least two displays. Therefore, this setup presented an ideal testbed for occlusion culling methods.

The test cases presented here are two extreme ones, one sparse rural environment - Ancient Olympia, as presented in the "Walk through Ancient Olympia" thematic show of FHW [Gai04a] – and a dense cityscape. Ancient Olympia consists of approximately $250 \cdot 10^3$ triangles, while the cityscape case uses about $500 \cdot 10^3$ triangles. The respective graphs in Fig. 5 and 6 show the number of primitives actually rendered (not culled) per walkthrough instance. For the statistic data capture, pre-recorded viewpoint position and orientation paths were used in order to maintain consistency among the experiments with different culling techniques.

In sparse environments, when the view position is far from the blocking geometry, the gain of the solid occluders over the planar ones is not significant. But when the viewpoint moves closer to objects, the solid occluders create larger, contiguous frusta, which effectively cull the hidden objects. Fig. 5 demonstrates this fact. At the view locations marked as A, B and C, the camera is facing towards the adjacent buildings. As planar occluders fail to cover both visible sides of a building simultaneously, hidden geometry is partially occluded, while in the case of the solid occluders it is fully occluded. Solid occluders are very convenient for structural geometry, like buildings, where convex polyhedra can be very intuitively placed or generated to tightly match the structure's shape.

In the second case study presented here (Fig. 6), a camera navigates through a model city, viewing the scene from a wide range of angles and locations. In order to keep the number of occluders low, each building is associated with a single though sub-optimal occluder. Using three box occluders instead of one, would have resulted in a more tight fitting of the geometry and therefore better culling results. The scene complexity of modeled environment is very balanced with regard to the view-direction due to the uniform distribution of three-dimensional models in space. As can be verified by the measured visible primitives at each path location, in this dense environment, there is an almost constant performance ratio between the solid and the planar occlusion culling method.

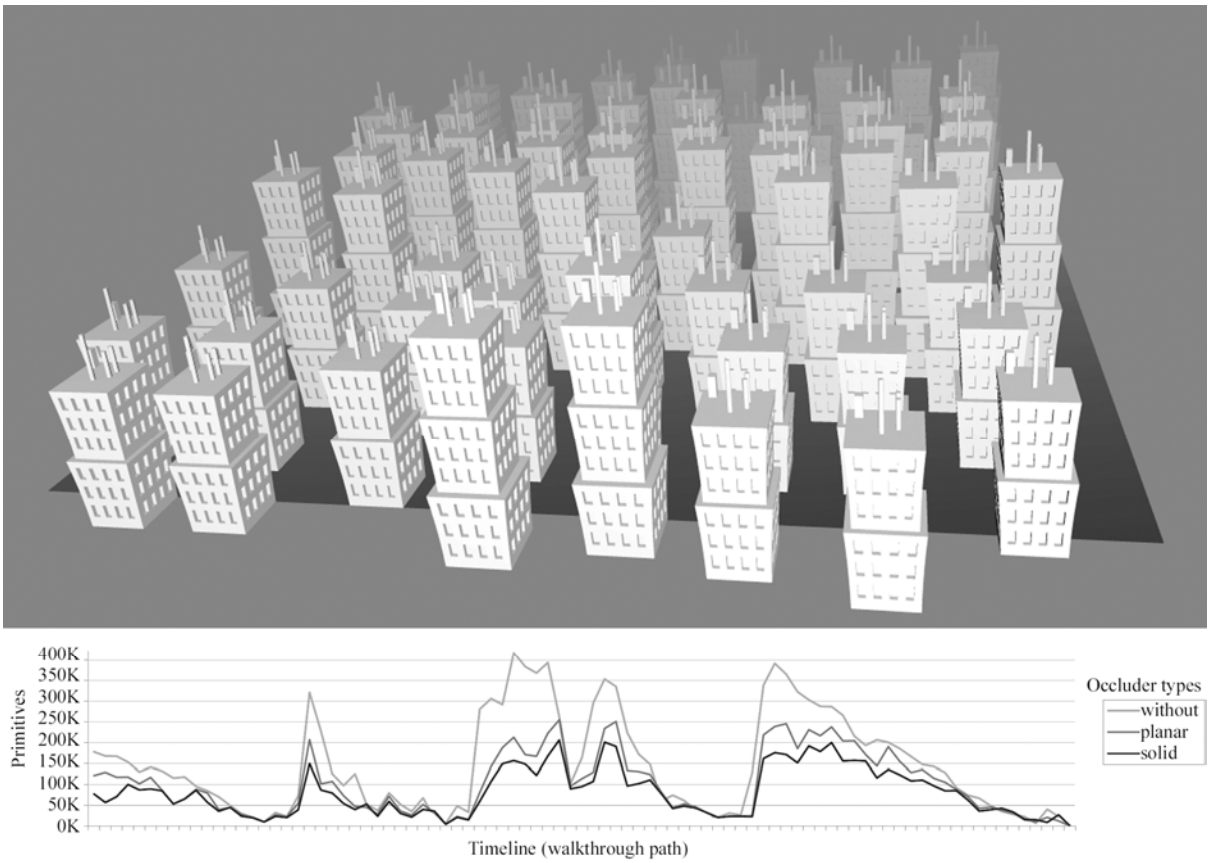

Figure 6. Dense environment test case. The chart shows primitives actually rendered (no occlusion / view frustum culling). There is an almost constant gain of the solid occluders over the planar ones.

Apart from the computational cost of the geometry-frustum containment test, which is common to both planar and solid occluder methods, the overhead introduced by the solid occluders comes from the determination of the visible edges of a convex solid, at each frame (see statistics in Table 1 and 2). In practice, this overhead is negligible because the silhouette is computed for very few and simple solid objects. This is ensured by the selection mechanism discussed in section 3.

The solid occluders culling method is an approximate algorithm, whose accuracy, compared to exact visibility calculation methods, depends on the selection of the occluders. In general, as occluders are enclosed in the shell of the actual geometric entities, the method is conservative, i.e. visibility is overestimated.

5. CONCLUSION

Even though the speed of today's graphics cards is increasing at a phenomenal rate, it will always be the case that researchers and game developers alike would like to push more geometry through the

Occlusion Method	No Occl. Culling	Planar Occluders	Solid Occluders
Average primitives/frame	73096.9	57150.1	52882.0
Average Draw Time (ms)	100.57	89.71	85.14
Average Overhead (ms)	-	0.99	1.89

Table 1. Ancient Olympia Statistics

Occlusion Method	No Occl. Culling	Planar Occluders	Solid Occluders
Average primitives/frame	145488.8	100038.2	81064.2
Average Draw Time (ms)	136.80	92.60	78.26
Average Overhead (ms)	-	1.19	2.63

Table 2. Cityscape Statistics

graphics pipeline than the card can handle at acceptable frame rates. Removing non-visible geometry before it reaches the pipeline is an important aspect of real time graphics. Occlusion culling in general and more specifically the shadow frustum culling method does exactly that, removing large portions of the scene-graph from the pipeline at a significant speed gain as we saw from the plots. Furthermore, the use of convex solid occluders pushes the rendering speed higher and provides an intuitive extension to the shadow frustum culling method.

One aspect that has not been investigated in this paper is the generation of frustums from non-convex solids. The silhouette determination of concave solids and the intersection of arbitrary geometry with a non-convex frustum is CPU intensive and well beyond the practical limits of real-time applications.

REFERENCES

[Coh03a] Cohen-Or, D., Chrysanthou, Y., Silva, C. T., and Durand, F. A Survey of Visibility for Walkthrough Applications, IEEE Trans. Visualization and Computer Graphics, 9, pp. 412-431, 2003.

[Coo97a] Coorg, S., and Teller, S. Real-time Occlusion Culling for Models with Large Occluders, in ACM Symposium on Interactive 3D Graphics proc., pp. 83-90, 1997.

[Dow01a] Downs, L., Möller., T., and Séquin, C. H. Occlusion Horizons for Driving through Urban Scenery, in ACM Symposium on Interactive 3D Graphics proc., pp. 121-124, 2001.

[Gai04a] Gaitatzes, A., Christopoulos D., and Papaioannou, G. The Ancient Olympic Games: Being Part of the Experience, in Eurographics 5th International Symposium on Virtual Reality, Archaeology and Intelligent Cultyral Heritage (VAST 2004) proc., pp. 19-28, 2004.

[Gre93a] Greene, N., Kass, M., and Miller, G. Hierarchical Z-Buffer Visibility, in ACM SIGGRAPH ’93 conf. proc., pp. 231–240, 1993.

[Hud97a] Hudson, T., Manocha, D., Cohen, J., Lin, M., Hoff, K., and Zhang, H. Accelerated Occlusion Culling Using Shadow Frustra, in 13th Annual ACM Symposium on Computational Geometry proc., pp. 1–10, 1997.

[Kol00a] Koltun, V., Chrysanthou, Y., and Cohen-Or, D. Virtual occluders: An Efficient Intermediate PVS Representation, in Rendering Techniques 2000: 11th Eurographics Workshop on Rendering proc., pp. 59–70, 2000.

[Lue95a] Luebke, D., and Georges, C. Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets, in ACM Symposium on Interactive 3D Graphics proc., pp. 105–106, 1995.

[Sch00a] Schaufler, G., Dorsey, J., Decoret, X., and Sillion, F. Conservative Volumetric Visibility with Occluder Fusion, in ACM SIGGRAPH 2000 conf. proc., pp. 229–238, 2000.

[Tel91a] Teller, D., and Sequin, C. H. Visibility Preprocessing for Interactive Walkthroughs, in ACM SIGGRAPH '91 conf. proc., pp. 61-69, 1991.

[Zha98a] Zhang, H. Effective Occlusion Culling for the Interactive Display of Arbitrary Models, Ph.D. Dissertation, Department of Computer Science, UNC-Chapel Hill, 1998.