Virtual Multiresolution Screen Space Errors: Hierarchical Level-of-Detail (HLOD) Refinement Through Hardware Occlusion Queries

Jean Pierre Charalambos
Departament de LSI, Universitat Politecnica de Catalunya, Barcelona, Spain
charalam@lsi.upc.edu

Abstract

We present a novelty metric to perform the refinement of a HLOD-based system that takes into account visibility information. The information is gathered from the result of a hardware occlusion query (HOQ) performed on the bounding volume of a given node in the hierarchy. Although the advantages of doing this are clear, previous approaches treat refinement criteria and HOQ as independent subjects. For this reason, HOQs have been used restrictively as if their result were boolean. In contrast to that, we fully exploit the results of the queries to be able to take into account visibility information within refinement conditions. We do this by interpreting the result of a given HOQ as the virtual resolution of a screen space where the refinement decision takes place. Our new error metric is general enough to be employed in any HLOD-based system as the quantity that guides its refinement. Despite its simplicity, in our experiments we obtained a meaningful performance boost (compared to previous approaches) in the frame-rate with almost no loss in image quality.

1. Introduction

Visualization of complex models comprising several million polygons is a very active research area [13, 5, 8, 15, 3, 7]. For the interactive rendering of such models, hierarchical level of detail (HLOD) methods have proven to be the most efficient approach. Typically, a HLOD-based system is built performing two recursive offline steps: 1. Without assuming anything about the topological genus of the underlying model, a scene hierarchy is built in a top-down manner using spatial subdivision driven solely by a user-defined target number of primitives per node. 2. For each node of this hierarchy a single, or a few levels of detail are generated using offline simplification in a bottom-up manner. HLODs support out-of-core algorithms in a straightforward way, and allow an optimal balance between CPU and GPU load during rendering [8]. The HLODs either consists of a point-based [13], or polygon-based approximation of the model [5].

During runtime visualization of the model two of the main tasks needed to be accomplished are: the refinement of the HLOD hierarchy and performing the occlusion culling. The quantity that has traditionally driven the refinement of the hierarchy is the screen projection error. This quantity corresponds to the number of pixels obtained when projecting onto the screen a given model space error related to a node (see Section 2.1). The preferred method of performing the occlusion culling has been by means of hardware occlusion queries (HOQs). To test the visibility of a node, a HOQ is issued on the node bounding volume. The returning value of the test corresponds to the precise number of pixels of the volume that would result as being visible (see Section 2.2).

Occlusion culling offers an excellent measurement for HLOD refinement criteria. According to the degree of partial occlusion of a node, it could be determined that there would be no gain in the final appearance of the image obtained if the node were further refined. However, to our knowledge there are no HLOD approaches which take visibility information as an integral part of the refinement condition. With this purpose in mind, in this paper we provide the means to relate the previously stated refinement criterion with HOQs (see Section 3). The novelty of our approach is that we do this by interpreting the result of a given HOQ as the virtual resolution of a screen space where we are going to project a given model space error. The virtually switched screen space error obtained in this way is then used as the quantity that guides refinement, effectively taking into account the result of the HOQ (see Section 4).

Some of the properties of our new HLOD refinement metric are: 1. Improved performance with the same visual quality: we are able to render less primitives with almost no loss in image quality (see Section 5); 2. Generality: our metric supports both polygon-based as well as point-based HLODs. 3. Full use of the result of the HOQ: our metric takes full advantage of the information gathered in HOQs; and, 4. Straightforward implementation.
2. Related work

2.1. Refinement criteria: model and screen projection errors

Hierarchical Refinement. The screen projection error is the quantity that has traditionally driven the refinement of a HLOD-based system. In a top down traversal of the hierarchy the refinement condition may be written as follows: if \((\varepsilon \leq \tau)\) then stop hierarchical refinement (e.g., [3]), where: \(\varepsilon\) corresponds to the screen projection error, i.e., the projection onto the screen of a model space measurement \(\lambda\), and \(\tau\) corresponds to the user specified threshold given in pixels (sometimes known as pixels of error [15]).

In polygon-based HLODs, \(\lambda\) corresponds to the model space error due to the simplification of the geometry related to a given node in the hierarchy. Readers may refer to Lindstrom [11], for a possible estimation of \(\lambda\) from the quadric error metrics [6]. An upper bound of \(\varepsilon\) could be obtained by measuring in pixels the projected diameter of a sphere with diameter equal to \(\lambda\) and centered at the node bounding sphere point closest to the viewpoint [3] (see Figure 1).

![Figure 1. Screen Projection Errors in Polygon-Based HLODs.](image)

In point-based HLODs, \(\lambda\) corresponds to the bounding volume of a given node, e.g., the node bounding sphere [13], or the node bounding box [7].

Refinement within the nodes. During runtime inspection of the model, in polygon-based HLODs popping artifacts may appear when refining (or coarsing) in a given frame a previous visible node. Those visual artifacts are prominent in hierarchies with fast growing subdivision [12]. To avoid their appearance, instead of a single level of detail, a few levels of detail are related to each node. Thus, an additional finer-grained refinement needs to be achieved within the nodes of the hierarchy [15]. To perform this refinement a range of model space errors \([\lambda_{\text{min}}, \lambda_{\text{max}}]\), is first (during the simplification step) related to each node. When visiting a node during runtime, \(\tau\) is mapped back from screen to model space to obtain \(\lambda_{\tau}\), i.e., \(\lambda_{\tau}\) simply corresponds to the value of the model space error that when projected onto the screen leads to \(\varepsilon = \tau\). The value of \(\lambda_{\tau}\) is then used to: 1. Refine the hierarchy by means of the following refinement condition: if \((\lambda_{\text{min}} \leq \lambda_{\tau} \leq \lambda_{\text{max}})\) then stop hierarchical refinement. 2. Refine the geometry within the node. Readers may refer to Hoppe [10] for the details of how to accomplish the refinement of a geometric model when \(\lambda_{\tau}\) is given.

2.2. Hardware occlusion queries

The HOQ scan converts a set of graphics primitives (but does not render them to screen), and determines whether or not any pixels in the frame buffer would be affected if the primitives were actually rendered to the screen [9]. HOQs have several advantages: generality of occluders, occluder fusion, generality of occludees, better use of GPU power, and easy use. For this reason, HOQs have become the preferred method in those HLOD-based systems implementing occlusion culling, e.g., [15, 7]. However, their main disadvantage is that there is a latency between issuing a query and the availability of the result [2].

Currently, the two main supported properties of HOQs in OpenGL (see [14]) are:

1. Multiple occlusion queries may be sent at once.
2. The returning value corresponds to the number of visible pixels of the queried object, but without telling anything about their position.

By means of the first property, authors have focused their attention on avoiding CPU stalls due to latency. An elegant and powerful method to minimize them was introduced by Bittner et al., [2]. The method fully exploits the spatial and temporal coherence of visibility, inherent to hierarchical representations. This strategy could easily be adapted to any HLOD-based system, as it is done in Gobetti et al., [7].

However, to our knowledge the second property has received little or no attention. In all previous HLOD approaches to test the visibility of a node in the hierarchy, a HOQ is issued on its bounding volume. If in its result there are no visible pixels, the object related to the node is culled away. If there is at least one visible pixel and if \((\varepsilon > \tau)\), then the node is further refined [15, 7], i.e., HOQs have been used as if their result were boolean. The main issue with this approach is that it results as being too conservative, particularly when the number of visible pixels is low.

3. Overview

Few approaches exist that integrate LODs with occlusion culling. A cell/object-precision occlusion culling method...
was introduced in Andujar et al., [1] (readers may refer to Cohen et al., [4] for a taxonomy of occlusion culling methods). The scene is divided into cells and for each cell the objects are classified into sets according to their visibility degree, i.e., a representative error that measures the possible contribution in pixels to the final image is assigned to each set. From the selection of some objects that act as occluders, the sets related to each cell are obtained during a preprocess. During runtime the error is used to: discard the objects of the sets not meeting a user defined threshold; or, for the objects having a multiresolution representation, to select the proper level-of-detail needed to display them. Unfortunately, this method is incompatible with HLODs, i.e., in HLODs we do not assume anything about the topological genus of the scene, and thus the occlusion method should be generic [4] (see Section 1).

Our idea of integrating occlusion culling directly into the refinement criterion of a HLOD-based system is based on the following observation. As previously stated, during rendering time, state-of-the-art hierarchical refinement corresponds solely to the comparison between a screen space error, given in number of pixels, and a user specified threshold (see Section 2.1). However, because of its advantages the preferred method of performing the occlusion culling is through HOQs (see Section 2.2). Nevertheless, the result of the queries is the number of visible pixels of a given object. Since occlusion culling also offers an excellent refinement criterion (see Section 1) it seems reasonable to integrate it into the above refinement condition. In this way we could expect to obtain an increment in the average rendering frame-rate: once we determine that there is no gain in further refining a node, due to partial occlusion, we could stop the refinement in advance, sending less primitives to the graphics card while achieving the same approximated visual quality.

Observe that a naive approach to using the result \( q \) of a HOQ, would be to redefine the refinement condition as follows: if \((\varepsilon \leq \tau) \text{ or } (q \leq \psi)\) then stop refinement, where \( \psi \) would be an additional user defined threshold given in pixels. The reason is that the new proposition \( q \leq \psi \), is viewpoint independent, i.e., it does not take into account the viewpoint, nor other viewing parameters; and thus it is incompatible with the nature of the refinement criterion.

4. Virtual multiresolution screen space errors

In contrast to previous approaches our novelty is that we employ the result of HOQs as an integral part of the HLOD refinement criterion. We interpret the result of a given HOQ as the virtual resolution of a screen space where we are going to project a given model space error \( \lambda \). In all previous approaches \( \lambda \) is directly projected onto the screen space \( S \), at its full original resolution (see Section 2.1). Similar to previous approaches we also use the value of the new error metric, dubbed virtual multiresolution screen space error, as the quantity that guides refinement.

Let \( M \) be the space on the screen defined as the subset of \( S \) that corresponds to the projection of a given node’s bounding volume \( B \). Observe that \( M \) bounds the projection onto \( S \) of the model space error related to \( B, \lambda \); and also the (eventual) projection of its related geometry \( G \). Let us say that the resolution of \( M \), denoted by \( \alpha \), corresponds to the number of pixels of the projection of \( B \) onto \( S \). Let \( q \) be the number of visible pixels on \( S \) obtained when a HOQ on \( B \) is issued, i.e., the result of the query; and \( q' \) be the number of invisible pixels. Observe that \( q' = \alpha - q \), and that \( 0 \leq q \leq \alpha \) always holds. Also note that, if in the moment when the query was issued there were no objects between \( S \) and \( B \), then \( \alpha = q \).

Suppose that the result of the HOQ performed on \( B \) is available, i.e., we know \( q \); and remember that we want to use its value as an integral part of the refinement criterion, i.e., we want to establish how \( q \) could affect \( \varepsilon \). However, in polygon-based HLODs \( \lambda \) is not subject to occlusion: for the sole purpose of projecting \( \lambda \) we always need to assume that all the pixels of \( M \) are visible (see Section 2.1). Our approach is therefore to coarsen the resolution of \( M \) by means of \( q \), i.e., we calculate from \( q \) a coarser virtual screen space resolution \( \delta \), \( \delta = f(q) \leq \alpha \), for the screen space where we are going to project \( \lambda \). The visual appearance obtained when projecting \( G \) due to \( \varepsilon \), could then be approximated using \( \delta \) in one of two ways:

1. Calculate \( \varepsilon \) in the original (real) resolution \( \alpha \), while projecting \( G \) at the coarse (virtual) resolution \( \delta \).
2. Calculate \( \varepsilon \) in the coarse (virtual) resolution \( \delta \), while projecting \( G \) at the original (real) resolution \( \alpha \).

In terms of the biased (see below) visual appearance obtained, the two statements above are almost equivalent. However, in the latter case we would be able to stop the refinement higher in the hierarchy allowing us to send less primitives to the GPU, i.e., effectively taking into account the result of the HOQ.

4.1. How much to coarsen a virtual resolution

To best decide how to calculate \( \delta \) it would be necessary to measure the introduced bias, i.e., the loss in image quality. However, to characterize the bias it would be necessary to know the exact position of the pixels comprising \( q \). Since this functionality is not currently present in HOQs (see Section 2.2) in this paper we simply assume that \( \delta = q \). That is, in order to project \( \lambda \) we coarsen the resolution of \( M \) from \( \alpha \) to \( q \). Observe that for extreme visibility conditions this assumption leads to reasonable results, i.e., if \( q = 0 \) then the
value of \( \varepsilon \) at the coarse virtual resolution would be 0; and if \( q = \alpha \) then the value of \( \varepsilon \) at the coarse virtual resolution would be the same as if no visibility information would have been gathered. Fortunately, for most practical purposes we have found this to be a good approximation. Most likely because the closer \( q \) gets to 0, the bias gets a higher chance to decrease (see Section 5).

4.2. Switching among multiple virtual resolutions

Since we cannot allow an actual change in the resolution to take place, we now seek to compute \( \varepsilon \) at a virtual resolution \( q \), from its computation at \( \alpha \). We can then take the value of the former to effectively refine the hierarchy, without actually modifying the original resolution. For this, we show how \( \varepsilon \) could be virtually switched among multiple resolutions.

We begin by extending the definition of the screen projection error according to a given screen resolution \( \gamma \). First suppose that when projecting \( \lambda \) onto the screen space, each of the obtained pixels occupy a position \((i, j)\). Let us say that the virtual multiresolution screen projection error \( \varepsilon^{\gamma} \) at a given screen resolution \( \gamma \), is given by the following expression:

\[
\varepsilon^{\gamma} = \sum_i \sum_j I^{\gamma}_{i,j}
\]

Where \( I^{\gamma}_{i,j} \) corresponds to the intensity value of the pixel at the screen position \((i, j)\), and at the given screen resolution \( \gamma \). Note that if \( \gamma = \alpha \), then \( \varepsilon^{\gamma} \) reduces to the original definition of \( \varepsilon \), i.e., in this case \( I^{\gamma}_{i,j} \) are 0 or 1, and thus \( \varepsilon^{\gamma} \) simply corresponds to the number of pixels of the projected error.

Now let \( a^{\gamma} \) be the area occupied by a single pixel at a given screen resolution \( \gamma \). To keep the estimation of the error consistent when virtually switching the resolutions, the total intensity due to \( \varepsilon^{\gamma} \) should be maintained constant, i.e., this will produce exactly the same effect on \( \varepsilon \), as if we were to actually change the screen resolution. Therefore, since the total intensity due to \( \varepsilon^{\gamma} \) is given by \( a^{\gamma} \sum_i \sum_j I^{\gamma}_{i,j} \), it follows that:

\[
a^{\gamma} \sum_i \sum_j I^{\gamma}_{i,j} = a^{\alpha} \sum_i \sum_j I^{\alpha}_{i,j}
\]

However, since \( a^{\gamma} \times q = a^{\alpha} \times \alpha \) and \( \varepsilon = \varepsilon^{\alpha} \), we finally get:

\[
\varepsilon^{q} = \varepsilon \times (q/\alpha)
\]

Thus, our hierarchical refinement condition that takes into account HOQs may be simply written as: if \( ((\varepsilon \times (q/\alpha)) \leq \tau) \) then stop hierarchical refinement.

4.3. Estimation of the number of pixels obtained when projecting a node bounding box

It only remains to show how \( \alpha \) could be calculated. The exact computation of \( \alpha \) (see the previous observation regarding when \( \alpha = q \)) suffers from a double drawback: we would need to issue an additional HOQ on a reset z-buffer. While there are several ways to approximate the value of \( \alpha \), our approach is to do it as follows. As a part of the preprocessing: 1. We first build a cube \( \bar{B} \) with the same volume as \( B \). 2. On an orthogonal view direction from the center of one of the faces of \( \bar{B} \), we calculate the shortest distance \( \widehat{d} \), at which the cube is completely visible and count the number of pixels \( \widehat{f} \), of the face. 3. We keep at the node the values of \( \widehat{f} \) and \( \widehat{d} \). It is easy to see that the estimated value of \( \alpha \), to be used at runtime, is then simple given by:

\[
\widehat{\alpha} = \widehat{f} \times (\widehat{d}/\bar{d})^2
\]

Where \( d \) is the same distance used to calculate \( \varepsilon \), i.e., the distance from the viewer to the center of the node’s bounding sphere, at a given frame. Observe that the above estimation leads almost always to an underestimated value of \( \alpha \), i.e., otherwise we would be underestimating the value of \( \varepsilon^{q} \), something that is better avoided.

4.4. Refinement within the nodes

So far we have only focused our attention on hierarchical refinement. However, we can easily adapt our above framework for those hierarchies where it is necessary to further refine the model within the nodes (see Section 2.1). Since the user specified threshold is given in screen space, we can use the result of the query to adjust \( \tau \) exactly in the same way as we did with \( \varepsilon \), i.e., we can simply: 1. Compute \( \tau^{q} \) as \( \tau^{q} = \tau \times (q/\alpha) \); 2. Compute \( \lambda_{\tau} \) by mapping back \( \tau^{q} \) to the model space; and, 3. Use \( \lambda_{\tau} \) to refine the model.

5. Results

An experimental software supporting our new metric has been implemented on Linux using C++ with OpenGL. To evaluate the performance boost and the image quality obtained when using \( \varepsilon^{q} \) with respect to \( \varepsilon \), we have extensively tested our system with a number of scenes with different depth complexities. We have implemented a geometry-based HLOD with an oct-tree with nearly 2000 triangles per node. We have employed the quadric error metrics (see [6]) to simplify the geometry, and also to derive the model space errors [11]. To refine the hierarchy we have made use of a top-down/front-to-back traversal algorithm based on bit toggling [9]. All tests were run on a window size of
640*480 and \( \tau = 1 \), and on a Pentium M 1.7GHz with a nVidia GeForce Go 6200.

For our discussion in this paper we built two scenes with middle and high depth complexities, respectively named as scene 1, and scene 2 (see Figures 6 and 7). For each scene we have designed a session representing typical inspection tasks. Our inspection sequences include rotations and changes from overall views to extreme close-ups that heavily stress the system. For each scene we first used \( \varepsilon \) to refine the hierarchy and then played the sequence to collect data: frame rates and number of drawn nodes. Afterwards the sequence was replayed using \( \varepsilon^q \) instead of \( \varepsilon \).

**Performance boost.** In scene 1 we have obtained an average of 25% of performance boost in the frame rate when refining the hierarchy using \( \varepsilon^q \) in respect to \( \varepsilon \). Figure 2 shows the whole frame rate values of the inspection sequence. The main reason for this improvement was that we were able to send less primitives to the GPU. We saved an average of 21% of the total number of nodes that needed to be rendered. Figure 3, shows the whole sequence of drawn nodes. For scene 2 we achieved more impressive results. We have obtained an average of 46% of performance boost in the frame rate when refining the hierarchy using \( \varepsilon^q \) in respect to \( \varepsilon \). Figure 4 shows the whole frame rate values of the inspection sequence. We saved an average of 35% of the total number of nodes that needed to be rendered. The whole sequence with the number of drawn nodes is depicted in Figure 5.

In general, we have found that the higher the depth complexity of a scene is our metric performs better. In scenes with a low depth complexity our metric performs at least as well as in previous approaches.

**Image quality.** Figures 6 and 7 show the image quality of both scenes obtained in a close-up frame of the inspection sequence when using \( \varepsilon \) and \( \varepsilon^q \). Observe that the difference in the quality obtained between the final images is almost imperceptible. To illustrate the effect of the degree of correction of \( \varepsilon \) when refining the hierarchy using \( \varepsilon^q \), the view-frustum view of the bounding boxes of the nodes selected to be drawn are shown in the top row of both images. Moreover, when using \( \varepsilon^q \) we coloured the node bounding boxes from blue to magenta to red according to the degree of correction of \( \varepsilon \): blue means low correction \( (\varepsilon^q \gg \varepsilon) \), magenta means middle correction \( (\varepsilon^q < \varepsilon) \) and red means high correction \( (\varepsilon^q \ll \varepsilon) \), see the top of Figures 6(b) and 7(b). Because of the visibility information gathered with HOQs, observe that the errors of the nodes near to the viewpoint get lower correction, than those that are farther. Therefore the introduced bias in the final image due to the correction of \( \varepsilon \) gets alleviated, i.e., the higher the correction of \( \varepsilon \) is, the harder the chance that the geometry related to the node would result as being visible.

**6. Conclusions**

We have introduced a HOQ-based refinement metric that supports both polygon-based as well as point-based HLODs. The main contributions of our approach are:
Figure 5. Drawn nodes for scene 2.

- Improved performance with the same visual quality: we are able to render less primitives with almost no loss in image quality.

- Full use of the result of the HOQ: our metric takes full advantage of the information gathered in HOQs.

- Straightforward implementation.

For future work we are researching how to integrate other view-dependent parameters (different from the distance to the viewpoint) into the refinement criterion. We are also investigating means to integrate our introduced metric with a strategy to avoid CPU stalls due to the latency inherent to HOQs.

7. Acknowledgements

We would like to thank the Stanford Graphics Group for the 3D models they provided. We also thank Eduardo Romero at the Universidad Nacional de Colombia for his valuable assistance throughout the project. This work is funded by a Fundacion Carolina grant.

References


Figure 6. Scene 1 (middle depth complexity): Selected frame of the visualization sequence. The top images correspond to the view-frustum view of the bounding boxes of the nodes selected to be drawn.

Figure 7. Scene 2 (high depth complexity): Selected frame of the visualization sequence. The top images correspond to the view-frustum view of the bounding boxes of the nodes selected to be drawn.