Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Real-time appearance preserving out-of-core rendering with shadows

Michael Guthe, Pavel Borodin, Ákos Balázs, and Reinhard Klein

Institute of Computer Science II, Computer Graphics Group, University of Bonn, Germany

Abstract

Despite recent advances in finding efficient LOD-representations for gigantic 3D objects, rendering of complex, gigabyte-sized models and environments is still a challenging task, especially under real-time constraints and high demands on the visual accuracy. The two general approaches are using either a polygon- or a point-based representation for the simplified geometry. With the polygon-based approaches high frame rates can be achieved by sacrificing the exact appearance and thus the image quality. Point-based approaches on the other hand preserve higher image quality at the cost of higher primitive counts and therefore lower frame rates.

In this paper we present a new hybrid point-polygon LOD algorithm for real-time rendering of complex models and environments including shadows. While rendering different LODs, we preserve the appearance of an object by using a novel error measure for simplification which allows us to steer the LOD generation in such a way that the geometric as well as the appearance deviation is bounded in image space. Additionally, to enhance the perception of the models shadows should be used. We present a novel LOD selection and prefetching method for real-time rendering of hard shadows. In contrast to the only currently available method for out-of-core shadow generation, our approach entirely runs on a single CPU system.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Shading, shadow and texture I.3.3 [Computer Graphics]: Viewing algorithms E.2 [Data Storage Representations]: Object Representation

1. Introduction

Modern 3D acquisition techniques allow to digitize 3D objects with very high accuracy. Digitizing human sized objects in the sub-millimeter range has become common, posing new challenges for the rendering of this data. First, due to the sheer size of models which is often in the range of several gigabytes, out-of-core rendering techniques are needed. Second, the quality of the rendering has to reflect the accuracy of the acquired data. Third, additional visual effects like shadows that are commonly used to improve the perception of objects in computer graphics should be applicable to these models. Last but not least even for such complex objects all these requirements should be fulfilled in real time.

To meet at least some of these challenges, many different approaches – like hierarchical geometry representations, point based rendering, visibility culling, and even image-based methods – for interactive rendering of complex models have been developed in the recent years. Most of these algorithms have in common that additional data structures, like level of detail (LOD) structures, multi-resolution representations, images, or occluder information, have to be

Figure 1: *Images of the St. Matthew statue (372M triangles) rendered with state-of-the-art polygon based out-of-core rendering (left) and with our algorithm including soft shadows (right). Notice how the left picture appears blurred due to the lack of fine details.*

stored that further increase the memory requirements. While this already leads to problems with medium sized objects, many of these algorithms cannot be used for gigabyte-sized models like the one shown in Figure 1. Therefore, current research efforts concentrate on the extension of available in-core approaches to out-of-core methods that allow to restrict the memory footprint at runtime. These extended methods only load the currently required parts of the model and additional data structures into main memory and employ prefetching techniques to prevent load stalls whenever interactivity is required.

For interactive rendering of complex models, hierarchical level of detail (HLOD) methods have proven to be the most efficient approach. A scene hierarchy is build using spatial subdivision and for each node of this hierarchy a single, or a few levels of detail are generated using offline simplification. HLODs support out-of-core algorithms in a straightforward way and allow an optimal balance between CPU and GPU load during rendering. The HLODs either consist of a point- [RL00] or polygon-based [EM00] approximations of the model. While polygon-based HLODs lead to a higher performance especially for models with large smooth surfaces, the point-based HLODs preserve small features like wrinkles or chiselmarks much better. The reason for this is that in point-based approaches the geometry is tightly coupled to appearance attributes like normal and color, whereas in most polygon-based out-of-core simplification algorithms this coupling is neglected and therefore the polygon-based approximations tend to generate less primitives but destroy the appearance of the model. A further disadvantage of polygon-based approaches is that the continuity along the node boundaries has to be maintained explicitly.

Generating shadows for out-of-core models requires an appropriate LOD selection for shadow casters and the rendering of the shadows themselves. Unfortunately, the computational overhead on the CPU of the only so far existing out-of-core hard shadow algorithm is high. Therefore, it is applicable with reasonable speed only on a multi processor system or a on small cluster. Since it does not guarantee a reasonable screen space error for the shadows, disturbing popping artifacts occur during movement.

Finally, there is no rendering algorithm to generate real-time, or at least interactive soft shadows for out-of-core models.

In this paper we present the first appearance preserving out-of-core rendering algorithm based on HLODs that combines point- and polygon-based representations. We developed a new LOD selection method to render pixel accurate hard shadows at real-time frame rates using perspective shadow maps. Furthermore, we modify our shadow caster LOD selection algorithm to generate realistic soft shadows at interactive frame rates using penumbra quads. The main contributions of our new algorithm over state-of-the-art out-of-core rendering are:

- An accurate high quality real-time rendering of out-of-core models with guaranteed visual error (see Figure 1).
- A point/polygon balancing technique that allows a transition in both directions.
- A LOD selection algorithm for pixel accurate real-time shadows on a single CPU system.
- A specialized LOD selection algorithm for interactive high quality soft shadows.

We applied our algorithm to several gigabyte-sized models composed of up to hundreds of millions of polygons. Our out-of-core rendering algorithm typically uses a memory footprint of some hundred megabytes depending on the memory available on the graphics card. For performance evaluation we compare our method with purely point-based as well as non-appearance preserving polygon-based out-of-core rendering techniques.

The rest of the paper is organized as follows: we give a brief overview of previous work in Section 2. After that we describe the construction of our HLOD hierarchy in Section 3. The rendering and shadow caster LOD selection algorithm is described in Section 4 and results are given in Section 5. Finally, in Section 6 we present our conclusions.

2. Previous Work

Our new algorithm is based on previous work in the area of out-of-core rendering, appearance preserving level-of-detail, hybrid point/polygon representations and real-time shadow algorithms. Therefore, we give a brief overview of previous work in these fields.

2.1. Out-of-Core Rendering

The problem of rendering gigabyte-sized models was first addressed in terrain rendering. To reduce the per-triangle computation cost on the CPU, pre-computed terrain patches are assembled during run-time to shift the bottleneck from the CPU to the GPU like [Ger] and the RUSTiC [Pom00] and CABTT [Lev02] data structures. These methods were further improved by representing the patches within the nodes as irregular triangulated patches in a quadtree [KS01] or a binary tree domain [CGG*03]. Unfortunately these algorithms cannot be used directly for arbitrary 3D models since they rely on a parametrization of the mesh which is only trivial for terrain models.

A number of techniques like indexing, caching and prefetching [DP, SCH*01] were developed to increase the performance for large environment walkthrough applications. Recently some algorithms combining level-of-detail and culling have been proposed like [BSGM02]. Since this approach only worked for models that could be loaded into memory, it has been extended to handle gigabyte-sized models by employing out-of-core techniques in [VM02]. However, the screen space error is relatively high since they do

not control the Hausdorff error during simplification. Since these algorithms are based on the segmentation of objects into smaller subparts, the simplification along cuts is constrained. These problems were solved for terrain rendering in [CGG*03] and for 3D models in [GBK03] by filling introduced cracks in screen space and restricting the Hausdorff distance during simplification.

To generate the needed HLOD hierarchy a number of out-of-core simplification algorithms for large models have been developed [BMR99, CMRS03, LS01, SG01]. The currently most efficient out-of-core simplification algorithm [ILGS03] uses processing sequences and out-of-core compression to simplify gigabyte-sized models within a few hours. The main drawback of all out-of-core simplification algorithms mentioned so far is that they do not use the Hausdorff error during simplification. This was solved by Borodin et al. [BGK03], but no algorithm supports appearance preserving simplification with guaranteed error tolerance.

2.2. Appearance Preserving LOD

In the field of appearance preserving level of detail, two main approaches exist. The first approach is to use textures to store the information required for correct shading, the so-called normal maps [COM98]. They can be used for efficient shading in software or on programmable graphics hardware [TCRS00]. Since normal maps require a large amount of memory, some quality preserving approaches to compress textures on polygonal models have been proposed e.g. [BTB02, SGSH02]. However, while the normal map texture size despite compression remains a problem in the context of out-of-core rendering, there is a further drawback: since for rendering efficient HLODs have to be generated using topology modifying simplification, consistent parameterizations for the normal map textures of subsequent HLODs are hard to generate.

The second, less memory intense and more general approach is to use appearance preserving level of detail. Garland et al. modified their error quadrics [GH97] to preserve color, texture coordinates and normals [GH98]. However, guaranteeing a certain error of the geometry or the appearance during rendering is not possible using these modified error quadrics. As a different error measure for appearance preserving out-of-core simplification, the curvature of the mesh can be used as in [Lin02], but like for the modified error quadrics no screen space error can be guaranteed for this method. Another approach used for view-dependent refinement of multi-resolution meshes was introduced by Klein et al. [KSS98] which is able to control the shading error by guaranteeing that for each point on the screen the distance to the next correctly shaded pixel is below a specified constant. Unfortunately, this method cannot be used for static LODs, since the error measure is view-point dependent and requires the exact position and orientation of the surface on the screen to be known. Furthermore, the derivatives are calculated in screen-space which make it unapplicable to precomputed static LODs.

A different approach is perceptually driven simplification (e.g. [WLC*03]). But again this method requires knowledge of all viewing parameters – even for its basic features that produce results similar to appearance preserving simplification – and additional movement information for velocity simplification. Finally, peripheral simplification even requires tracking of the users eye movements.

2.3. Hybrid Point Polygon Algorithms

While the quality of point based rendering methods is sufficient, the performance of point based rendering covering the whole range from very coarse up to the finest LOD is still too slow for gigantic models. To overcome this problem, hybrid models combining point and polygon based rendering have recently been introduced.

A hybrid point/polygon-based representation of objects was first used by the POP rendering system [CN01], which uses polygons at the lowest level only and a point hierarchy similar to QSplat [RL00] on higher levels. Simultaneously a method for hybrid point polygon simplification based on edge collapse operations was introduced in [CAZ01]. In this approach points are generated according to the error metric and the size of the triangle. This algorithm however, allows a transition only from polygons to points and not vice versa, and therefore, the transition point has a high impact on the efficiency of the simplification. Another approach starting with a point cloud representation of the model is PMR [DH02]. The point cloud is simplified using a feature-based simplification algorithm and a triangulation of this point cloud is generated for display at higher resolutions afterwards. During rendering points or triangles are selected for rendering depending on their screen size. This approach adjust the point/polygon balance to achieve maximum rendering performance, but due to the triangulation of the simplified points cloud, the efficiency of the simplification algorithm is reduced.

2.4. Shadow Algorithms

In the recent years several algorithms for interactive shadow generation using graphics hardware have been developed. The two basic approaches to this problem are shadow maps [Wil78] and shadow volumes [Cro77].

To reduce aliasing artifacts inherent in the image-based approach of the shadow map algorithm, the perspective shadow map [SD02] was developed which takes the perspective projection into account to generate a more evenly sampled shadow map.

Since many improvements have been made to the shadow volume algorithm we refer to [MHE*03] for details. Due to advances in recent graphics hardware developments,

shadow volume computations can be completely performed on the GPU [BS03]. Since the generation of these shadow volumes is still too slow for complex scenes, a hybrid shadow map/shadow volume algorithm has been developed [GLY*03] which combines the speed of the shadow map with the accuracy of the shadow volumes. However, due to the computational overhead of this algorithm it is only applicable on a multi processor system or a small cluster. In addition there is no control over the screen space error of the shadows which leads to popping artifacts during movement.

Although enhancing the visual appearance, the hard shadows produced by the methods mentioned above suffer from a lack of realism, since all natural light sources produce soft shadows which depend on the size and distance of the light source. Due to their higher computational complexity compared to hard shadows they are even more challenging in the context of gigabyte-sized models and have not been used for out-of-core rendering so far. A recent survey on soft shadow algorithms has been made by Hasenfratz et al. [HLHS03]. The first methods for interactive soft shadows were image based techniques like [ARHM00]. A straightforward approach is rendering the scene with several shadow maps and then combining the image to generate soft shadows e.g. on a cluster [ISH03]. For shadow maps the first real-time algorithm for a single GPU system was the penumbra maps [WH03]. Since this algorithm renders only the outer half of the soft shadow (and a full shadow inside), the visual quality can be improved by combining this method with the shadow map [KD03] which only renders the inner half of the soft shadow. Recently an algorithm capable of rendering both inner and outer penumbra at real-time frame rates for moderately complex scenes using penumbra quads [AW04] was developed. For higher quality and more precise soft shadow calculation, the shadow volume algorithm was modified by Assarsson et al. [ADMAM03]. Due to the limited performance of shadow volumes this is not usable for complex scenes.

Although these shadow rendering algorithms can also be used for out-of-core rendering, appropriate LODs have to be selected for the shadow casters. So far there is no explicit LOD selection and prefetching algorithm for out-of-core models that guarantees a pixel correct location for the shadow silhouettes. Furthermore, there is no LOD selection algorithm that exploits the special requirements and restrictions of soft shadows.

3. HLOD Generation

Since our HLOD generation algorithm starts with a polygonal representation of the model, a point cloud model has to be triangulated in advance using standard reconstruction tools like [CL96]. Then, we follow the out-of-core simplification of Borodin et al. [BGK03] building an octree based HLOD hierarchy.

The partitioning algorithm starts with the whole model in the root node of an octree. The object is partitioned by cutting the geometry contained in each node into the eight child nodes and storing them in its children. This is repeated recursively until each leaf node contains at most T_{max} triangles. Starting from the geometry contained in the leaf nodes the HLOD hierarchy is build recursively from bottom to top with the following algorithm:

- Gather the simplified geometry from all child nodes that are two levels below the current node (or the original geometry if there is no HLOD at this depth). Its approximation error ε_{prev} is then the maximum error of the simplified geometry in these child nodes.
- Simplify resulting geometry as long as the simplification error is less than $\varepsilon_{simp} = \frac{e_{node}}{res} - \varepsilon_{prev}$, where e_{node} is the edge length of the currents nodes bounding cube and *res* is the desired resolution in fractions of e_{node}.
- Store $\varepsilon_{node} = \varepsilon_{simp} + \varepsilon_{prev}$ as approximation error in the current node.

This algorithm guarantees a Hausdorff distance ε_{node} of the simplified geometry to the original model to lie between $\frac{3}{4}\frac{e_{node}}{res}$ and $\frac{e_{node}}{res}$ for a user specified resolution *res*. Additionally, it closes the cracks introduced by the hierarchical simplification using generalized pair contractions [BGGK03]. But in contrast to the previous out-of-core rendering method [GBK03], we do not use the Hausdorff distance between the simplified and the original geometry, but the novel appearance preserving error measure described in Section 3.1.

Finally the geometry of each node is compressed and stored on disk, as well as the skeleton of the scene graph containing the bounding boxes and the geometric error of the simplified geometry contained in each node.

3.1. Preserving the Appearance

To preserve the appearance of the object during simplification, we extend the geometric Hausdorff error measure with respect to appearance attributes as proposed by Klein et al. [KSS98] for view-dependent multi-resolution meshes. However, in contrast to this approach we need an error measure that is independent of the viewing position.

When an edge is removed due to a collapse operation, the appearance attributes of the removed points are interpolated during rendering. A screen space error can now be defined as the distance between a shaded point of the original model projected into screen space and the next pixel on screen with the same color. For static LODs this distance can directly be transformed into object space as the distance between a point on the approximated surface and the next point on the original mesh with the same appearance attribute.

We define the simplification error in object space to be the distance of a point p on the original mesh and the closest point on the simplified mesh with the same interpolated normal q (see Figure 2). Now we make the observation that the

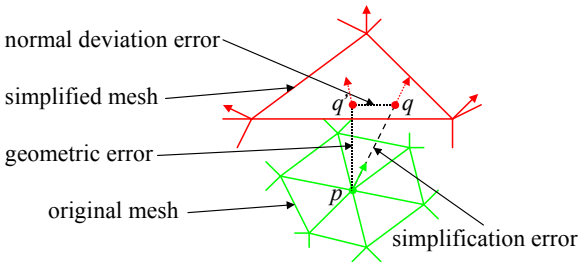

Figure 2: *Combination of error measures.*

vector between the original point p and q can be split into the orthogonal vectors pq' and $q'q$, where q' is the closest point on the simplified mesh. Therefore, the simplification error ε can be written as a combination of the geometric Hausdorff error ε_{geo} and the normal deviation error on the simplified mesh ε_{app}:

$$\varepsilon^2 = \varepsilon_{geo}^2 + \varepsilon_{app}^2$$

The normal deviation error ε_{app} can be approximated using the maximum normal curvature κ_1:

$$\varepsilon_{app} \approx \frac{\arccos(\vec{n} \cdot \vec{n}_{int})}{\kappa_1},$$

where $\vec{n}_{int}$ is the interpolated normal at q'. The maximum curvature of a point on a bi-linearly interpolated triangular patch with specified per vertex normals can be approximated by:

$$\kappa_1 \approx \max\left(\frac{\arccos(\vec{n}_1 \cdot \vec{n}_2)}{\|P_1 - P_2\|}, \frac{\arccos(\vec{n}_1 \cdot \vec{n}_3)}{\|P_1 - P_3\|}, \frac{\arccos(\vec{n}_2 \cdot \vec{n}_3)}{\|P_2 - P_3\|}\right).$$

For small angles, the computation of the inverse cosine can be saved, since in this case $\arccos(\vec{n}_a \cdot \vec{n}_b) \approx \|\vec{n}_a - \vec{n}_b\|$.

To prevent aliasing artifacts in the shading, we smooth the normals of vertices that are only adjacent to triangles smaller than $\frac{e_{node}}{res}$ before simplification. This also leads to a more efficient simplification.

While we only use normals in our examples, our algorithm is able to deal with arbitrary appearance attributes for which a distance is defined, e.g. per vertex colors, BRDFs, etc.

3.2. Point Generation

During the octree construction, triangles are allowed to become arbitrarily small. For rendering purposes however, using points instead of small triangles has proven to increase the performance significantly. Therefore, after finishing the complete hierarchical simplification the individual HLODs are processed and small triangles are replaced with points.

To find an appropriate criterion for the transition point between triangles and points we make the following observation: on modern graphics hardware a point – using the GL_POINTS primitive – can be rendered about twice as fast as a pixel sized triangle. Since according to Euler's formula, the number of vertices in a mesh is approximately half the number of triangles ($N_t \approx 2N_v$), not more than 3 additional points per vertex can be used without reducing the rendering performance.

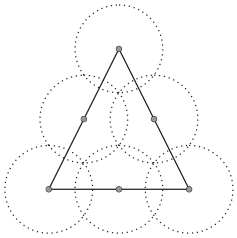

Figure 3: *Points used to replace a triangle.*

To determine which triangles are to be replaced with points, we simply check if the distance of the triangle vertices to the barycenter is at most than 2ε pixel on screen (typically: $\varepsilon = 0.5$). If this is the case, the points shown in Figure 3 cover the whole area of the triangle and we replace the triangle with up to 6 vertices. To avoid unnecessary points we use vertex clustering with a grid size of ε. During this vertex clustering the attributes are averaged similarly to [RL00].

This way the number of points used per HLOD is optimally adapted to the features of the simplified object. As shown in Table 2 it might even happen that it decreases with the coarser level, which of course would not be possible by simple clustering.

The main advantage of our technique is that due to the maximum size of a node on screen we can calculate the maximum number of pixels a triangle can cover. Therefore, we can apply the triangle to point transition during the preprocessing and store the points in the HLOD representation.

3.3. Compression

Even though prefetching is used (see Section 4.3) to predict which parts of the model will be visible next, the loading of parts from out-of-core devices (such as disc drives or over the network) is time-critical and can be a bottleneck. In order to minimize the resulting latency we employ sophisticated compression schemes capable of real-time decompression of the out-of-core data. As we use a hybrid point-polygon approach, we employ two well known compression schemes. To compress the triangle mesh we apply the Cut-Border algorithm for non-manifold meshes [Gum99]. Similar algorithms like Edgebreaker [Ros99] would work as well. In order to sufficiently compress the point data, we utilize the approach of Botsch et al. [BWK02].

4. Rendering

To render the scene we first determine the required level of detail and the visibility of cells. The octree is traversed and at each node the visibility is checked using view frustum and backface culling. For each visible cell its approximation error is projected onto the screen and if this screen space error is not sufficient the traversal is continued to finer levels.

The cracks introduced due to the independent simplification of adjacent cells need to be filled during rendering. In [GBK03] this was accomplished by rendering shaded billboard lines along these cracks. Since we guarantee a screen space error of at most 0.5 pixel for high quality rendering anyway, we can render simple lines instead which is more efficient.

4.1. LOD Selection for Shadows

Independently of the algorithm used to generate the shadow effect, the parts of the scene casting shadows have to be determined and an appropriate level of detail has to be selected.

For point or directional light sources, the level of detail required for a shadow caster depends on quantities shown in Figure 4 (left). Unfortunately, for shadows the approximation error depends not only on the distances between light source, caster and receiver but also on the angle of the incoming light and the surface normal of the receiver. If the incoming light is nearly perpendicular to the surface normal even the slightest change of the caster position leads to an arbitrarily high change in the shadow location on the receiver. Fortunately, this is only a problem if the receiver is highly specular since in all other cases the surface does not receive much irradiance from the respective light source. Therefore, by guaranteeing an accuracy for the shadow location for cases where the surface normal is nearly parallel to the incoming light direction to be better than 1/2 a pixel we inherently guarantee the accuracy to be better than 1 pixel in image space even for an angle of 60° between surface normal and incoming light. Please note that this angle on the other hand leads to a decrease of the irradiance from this light source by a half.

From Figure 4 (left) we get the following maximum approximation error ε_h of the shadow caster depending on the desired approximation error ε_r of the corresponding shadow receiver:

$$\varepsilon_h = \frac{\varepsilon_r d_l}{d_l + d_p}$$

Note, that for directional light sources (i.e. $d_l = \infty$), ε_h equals ε_r.

When using an area light source, the required approximation accuracy for a shadow caster depends on the relations shown in Figure 4 (right). If we allow an intensity change of γ_i, this leads to the following projected approximation error

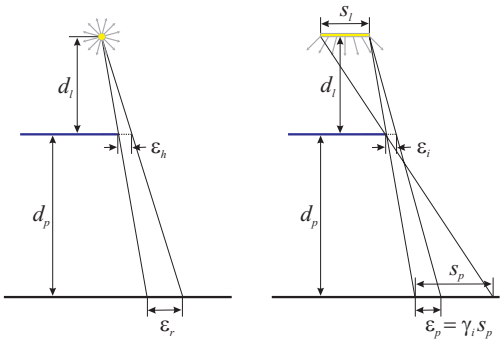

Figure 4: *Shadow caster approximation error for hard shadows (left) and soft shadows (right).*

ε_p:

$$\varepsilon_p = \gamma_i s_p = \gamma_i \frac{s_l d_p}{d_l}$$

The corresponding approximation error ε_i is the back-projection of this offset onto the shadow caster:

$$\varepsilon_i = \frac{\varepsilon_p d_l}{d_l + d_p} = \gamma_i \frac{s_l d_p}{d_l + d_p}$$

To combine these error measures, we simply add ε_h and ε_i. This means, that we allow the shadow edge to have an offset of at most ε_{screen} pixel offset on screen and an intensity change of at most γ_i. This is reasonable, since for hard shadows (i.e. a very small light source) ε_i has to be zero. So the maximum allowed approximation error ε_c for a shadow caster is:

$$\varepsilon_c = \frac{\varepsilon_r d_l}{d_l + d_p} + \gamma_i \frac{s_l d_p}{d_l + d_p} = \frac{\varepsilon_r d_l + \gamma_i s_l d_p}{d_l + d_p}$$

4.2. Shadow Generation

During rendering the coarsest possible LOD is chosen for the shadow casters. Since the appearance of an object is not relevant for shadow computation, a purely geometrically simplified HLOD representation can be used without loss of accuracy. The most general approach to effectively generate shadows on a single CPU are the so called shadow maps. This algorithm uses the hardware Z-Buffer to generate the required occlusion information. This technique however suffers from aliasing artifacts. To reduce artifacts that occur if the user zooms in (perspective aliasing) we use perspective shadow maps, where the scene is first transformed by the perspective view projection and then rendered from the position of the transformed light source.

Since the approximation error of all shadow receivers (i.e. all visible nodes) has to be known for both types of light sources, we first traverse the octree of the model and perform level of detail selection and culling. Then we compute

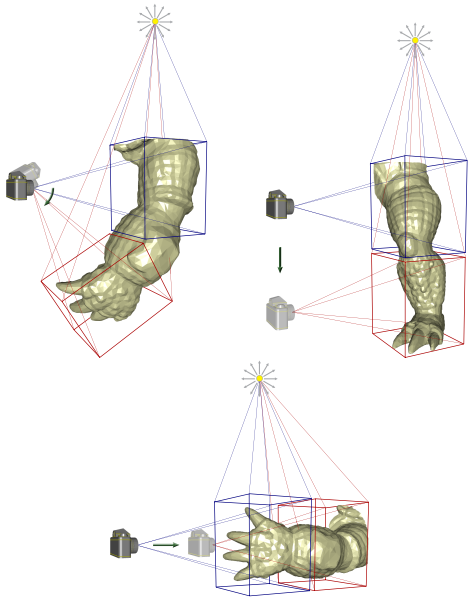

Figure 5: *Change of light sources view frusta due to rotation (top left) and translation (top right and bottom) of the viewer. To simplify matters only one light source frustum is shown.*

the view-aligned bounding box for each hierarchy level of the visible nodes. These bounding boxes are tight due to the relation of error and cell size and the regular octree structure. For each of these bounding boxes a minimum view frustum containing the whole box is calculated from the light source. These view frusta are then used for culling and level of detail selection of the shadow casters. To estimate the distance between a shadow caster and its first visible shadow receiver, we simply use the distance of the caster's cell to the current view frustum. This means that for all visible cells the shadow caster approximation error ε_c is always less or equal to its approximation error used for rendering ε_r. Therefore, the self-shadowing artifacts described in [GLY*03] cannot occur.

To render the shadow we use the perspective shadow map algorithm [SD02] with a sufficient resolution to guarantee at most 0.5 pixel screen space error for shadow boundaries (on a surface orthogonal to the light direction, see the above section). For soft shadows we use the penumbra quads [AW04] combined with perspective shadow maps. Since the shadow map generation only requires a geometric approximation of the model we do not use the appearance preserving simplification. This leads to a considerable speed-up of the shadow map generation.

4.3. Prefetching

Since geometry required for rendering must be streamed from disk, we use a priority based prefetching similar to [GBK03] in order to load data for subsequent frames. The loading priority of a cell's geometry depends on the viewer's movement that is necessary for the cell to become visible.

To support moving light sources, we apply the same priorities to the view frusta of each light to prefetch shadow caster geometry. Since both translation and rotation of the viewer result in rotations and zooms of the view frusta of the light sources as shown in Figure 5, a modification of these priorities is not necessary to support prefetching for a moving viewer.

5. Results

To analyze the efficiency of our approach, we compare it with several previous algorithms related to different aspects of our work, that can either guarantee a purely geometric screen space error or an appearance preserving screen space error. The models used for comparison are listed in Table 1. The system used for all performance evaluations is an Athlon 2800+ PC with a Radeon 9800 XP graphics card and 1 GB main memory.

model	#triangles	app. pres.	geometric
Dragon	871,414	4.9 MB	2.4 MB
Happy Buddha	1,087,716	7.5 MB	3.5 MB
David 2mm	8,254,150	35.5 MB	20.2 MB
Lucy	28,055,742	148.4 MB	66.7 MB
David 1mm	56,230,343	321.3 MB	138.6 MB
St. Matthew	372,422,615	1566.2 MB	649.6 MB

Table 1: *Triangle numbers of models used for testing and their size on disk for appearance preserving and non appearance preserving (geometric approximation) HLODs.*

5.1. HLOD Generation

Table 2 shows the average number of triangles and points in an octree cell for each level of the HLOD hierarchy. While the number of triangles per octree node of the purely geometrically simplified model is roughly constant, it varies strongly in the appearance preserving model due to variations in model features. Note that the transition from points back to triangles is clearly visible between level 1 and 0.

Of course due to the much more restrictive error measure the simplification rates become less compared to a pure geometric simplification. This overhead, however, is more than compensated for by the noticeable improvement of the object appearance.

For the finer levels of detail, the geometric error dominates the total simplification error. Therefore, the difference between appearance preserving and geometric simplification decreases. When the approximation error becomes higher, the relative curvature increases which leads to a much higher

LOD	#triangles	#points	#tri. (geom.)
0	3	368	107
1	0	1,429	334
2	71	5,636	597
3	3,208	22,255	955
4	31,073	72,151	1,313
5	48,369	122,117	1,672
6	19,174	55,278	2,030
7	17,170	42,041	2,384
8	12,170	22,759	2,089
9	10,203	12,984	1,791
10	7,703	3,969	1,492
11	5,092	756	1,204
12	2,052	112	835
13	896	50	478
14	1,379	0	895
15	1,194	0	1,194

Table 2: *Average triangle and point numbers of the octree cells in different LOD levels for the appearance preserving St. Matthew model compared to the triangle count of the purely geometrically simplified version.*

number of triangles and points compared to pure geometric simplification. At the coarsest LODs however, as the features are blurred by the normal antialiasing and thus the approximation error is dominated by the geometric error again.

Note that a purely point based algorithm needs to generate at least the same number of points as our approach plus the points represented by triangles. Since all triangles that can be rendered as six points are removed, at least 3 new points would be required for each triangle. If we assume an average number of 5 points per triangle, then a purely point based approach would require up to 360,000 point per node.

The differences in disk space of the HLOD models are already shown in Table 1. The disk space required for the QSplat models is comparable to that of the geometric approximation and amounts to 63.4 MB for the Lucy model and 644.4 MB for the St. Matthew model.

5.2. Out-of-Core Rendering

First we compare our algorithm to state-of-the-art out-of-core rendering without appearance preservation [GBK03] and to the QSplat rendering system [RL00]. For all three algorithms we use the same camera path, display resolution (640×480 pixel) and the same screen space error (0.5 pixel). Table 3 shows the average and minimum frame rates for different models.

As can be seen from Table 3, in the case of smaller models the rendering performance of our approach is relatively close to the purely geometric simplification. This is mainly due to the fact that the middle section of the HLOD hierarchy (see Table 2) becomes smaller and thus the maximum number of primitives per node is not significantly higher than for the geometric simplification. Additionally due to the coarser sampling these models are smoother and have less details to preserve.

model	app. pres.	geometric	QSplat
Dragon	99 fps	122 fps	n.a.
Happy Buddha	81 fps	117 fps	$\sim$8 fps
David 2mm	64 fps	155 fps	n.a.
Lucy	55 fps	110 fps	$\sim$4 fps
David 1mm	57 fps	114 fps	n.a.
St. Matthew	53 fps	93 fps	$<$1 fps

Table 3: *Average frame rates for different rendering algorithms.*

Regarding the comparison to the QSplat algorithm, the remarkable performance gain of QSplat in the case of smaller models also results from the coarser sampling, since pixel accuracy cannot be reached during closeups, as the original scanning of these models is too sparse.

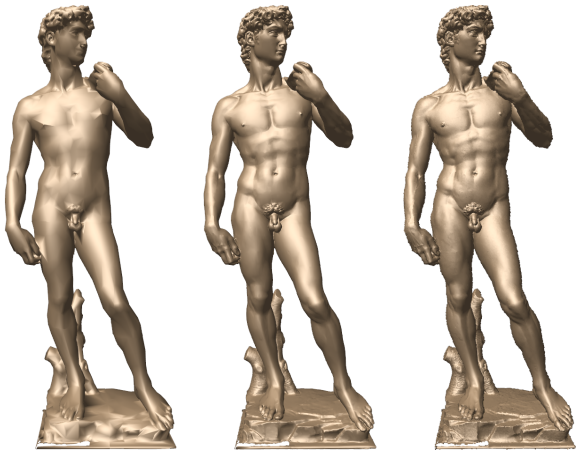

Figure 6: *David models rendered with purely geometric out-of-core rendering (left) and with our new appearance preserving algorithm (right). In the middle the model is rendered using geometric simplification but approximately the same number of primitives as for the right image.*

As shown in Figure 6 and the accompanying video, shading and popping artifacts that are visible for the purely geometry based simplification totally disappear with our approach since they fall below pixel scale. Although it is possible to reduce the artifacts of geometric simplification with a lower screen space error (see Figure 6 middle), small features are still blurred at the same frame rate and primitive count.

5.3. Shadow Generation

In Table 4 we compare the frame rates of our algorithm with different types of light sources, while Figure 7 shows the frame rates for the St. Matthew model with the three different light source types.

model	no shadow	point light	area light
Dragon	99 fps	75 fps	20 fps
Happy Buddha	81 fps	70 fps	20 fps
David 2mm	64 fps	55 fps	21 fps
Lucy	55 fps	40 fps	15 fps
David 1mm	57 fps	45 fps	17 fps
St. Matthew	53 fps	37 fps	13 fps

Table 4: *Average frame rates for different shadow algorithms.*

For hard shadows the frame rates are only reduced to 86% to 70% compared to our algorithm without shadows and are on average well above real-time. For soft shadows the generation of the inner and outer penumbra textures require two additional rendering passes with the pure geometrically simplified geometry. During each of these rendering passes approximately twice the number of primitives needs to be rendered to generate the penumbra quads. With respect to this much higher total primitive count, a drop to only 33% to 20% of the average frame rate is very good.

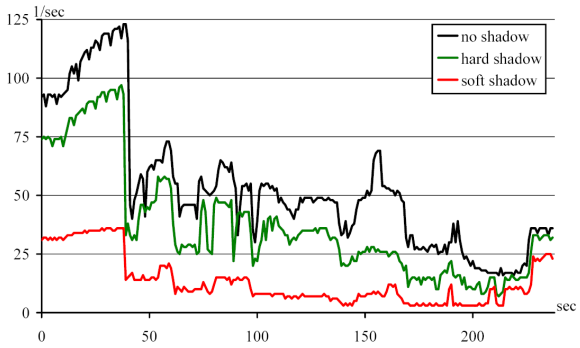

Figure 7: *Frame rate plot for the St. Matthew model with the three different types of light source for a recorded camera path. The first third is shown in the accompanying video.*

Figure 7 shows that the frame rate of our algorithm is always real-time even with hard shadows, except for sequences with very fast closeups like int the time between 190 and 230 seconds of the camera path. With soft shadows the frame rate is interactive to real time since it is always at least 3 frame per second and increases above 25 for distant views.

Finally, Figure 8 shows a screenshot from the camera path sequence with a point and an area light source.

Figure 8: *Screenshot from the camera path used for measurements with hard shadows (top) and soft shadows (bottom).*

Note that in contrast to [GLY*03] we do not need an additional PC in a cluster to compute the shadows, but achieve high quality hard shadows with only a minor overhead. Even soft shadows can be rendered at interactive frame rates with our method.

6. Conclusions

In this paper we presented an appearance preserving out-of-core rendering algorithm based on hierarchical levels of detail that combines point- and polygon-based approximations. To improve the comprehensiveness of the generated images,

we have developed a LOD selection method to render pixel accurate hard and soft shadows of moving light sources at interactive frame rates using perspective shadow maps and penumbra quads. The visual quality of the rendering is improved with only a small overhead compared to previous algorithms. Our appearance preserving error measure is general enough to be applicable to any type of surface attribute for which a distance measure can be defined, e.g. per vertex color, material or BRDF.

Acknowledgements

We thank Marc Levoy and the Digital Michelangelo Project for providing us with the models.

References

[ADMAM03] ASSARSSON U., DOUGHERTY M., MOUNIER M., AKENINE-MÖLLER T.: An optimized soft shadow volume algorithm with real-time performance. In *Siggraph/Eurographics Workshop On Graphics Hardware* (2003), pp. 33–40. 4

[ARHM00] AGRAWALA M., RAMAMOORTHI R., HEIRICH A., MOLL L.: Efficient image-based methods for rendering soft shadows. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 375–384. 4

[AW04] ARVO J., WESTERHOLM J.: Hardware accelerated soft shadows using penumbra quads. *Journal of WSCG 12*, 1 (2004), 11–18. 4, 7

[BGGK03] BORODIN P., GUMHOLD S., GUTHE M., KLEIN R.: High-quality simplification with generalized pair contractions. In *Proceedings of GraphiCon'2003* (September 2003), pp. 147–154. 4

[BGK03] BORODIN P., GUTHE M., KLEIN R.: Out-of-core simplification with guaranteed error tolerance. In *Vision, Modeling and Visualisation 2003* (November 2003), Ertl T., Girod B., Greiner G., Niemann H., Seidel H.-P., Steinbach E.„ Westermann R., (Eds.), Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 309–316. 3, 4

[BMR99] BERNADINI F., MITTLEMAN J., RUSHMEIER H.: Case study: Scanning michelangelo's florentine pieta. In *ACM SIGGRAPH 99 Course Notes Course 8* (1999). 3

[BS03] BRABEC S., SEIDEL H.-P.: Shadow volumes on programmable graphics hardware. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003), 433–440. 4

[BSGM02] BAXTER W. V., SUD A., GOVINDARAJU N. K., MANOCHA D.: Gigawalk: Interactive walkthrough of complex environments. In *Eurographics Workshop on Rendering* (2002), pp. 203–214. 2

[BTB02] BALMELLI L., TAUBIN G., BERNARDINI F.: Space-optimized texture maps. *Computer Graphics Forum (Eurographics 2002) 21*, 3 (2002), 411–420. 3

[BWK02] BOTSCH M., WIRATANAYA A., KOBBELT L.: Efficient high quality rendering of point sampled geometry. In *Eurographics Workshop on Rendering* (2002), pp. 53–64. 5

[CAZ01] COHEN J., ALIAGA D., ZHANG W.: Hybrid simplification: Combining multi-resolution polygon and point rendering. In *IEEE Visualization 2001* (2001), pp. 37–44. 3

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHINO F., SCOPIGNO R.: Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum 22*, 3 (2003), 505–514. 2, 3

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. *Computer Graphics 30*, Annual Conference Series (1996), 303–312. 4

[CMRS03] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R.: External memory simplification of huge meshes. In *IEEE Trans. on Visualization and Comp. Graph.* (to appear, 2003). 3

[CN01] CHEN B., NGUYEN M. X.: Pop: A hybrid point and polygon rendering system for large data. In *IEEE Visualization* (2001), IEEE. 3

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *Siggraph 1998, Computer Graphics Proceeding* (1998), Addison Wesley Longman, pp. 115–122. 3

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. *ACM SIGGRAPH Computer Graphics archive 11*, 2 (1977), 242–248. 3

[DH02] DEY T. K., HUDSON J.: PMR: Point to mesh rendering, a feature-based approach. In *IEEE Visualization 2002* (2002), pp. 155–162. 3

[DP] DECORO C., PAJAROLA R.: Xfastmesh: Fast view-dependent meshing from external memory. 2

[EM00] ERIKSON C., MANOCHA D.: HLODs for faster display of large static and dynamic environments. In *ACM Symposium on Interactive 3D Graphics* (2000). 2

[GBK03] GUTHE M., BORODIN P., KLEIN R.: Efficient view-dependent out-of-core visualization. In *The 4th International Conference on Virtual Reality and its Application in Industry (VRAI'2003)* (2003). 3, 4, 6, 7, 8

[Ger] GERSTNER T.: Multiresolution compression and visualization of global topographic data. SFB 256 report 29, Univ. Bonn, 1999 also in GeoInformatica, 7(1): 7–32, 2003. 2

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. *Computer Graphics 31*, Annual Conference Series (1997), 209–216. 3

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98* (1998), Ebert D., Hagen H.„ Rushmeier H., (Eds.), pp. 263–270. 3

[GLY*03] GOVINDARAJU N. K., LLOYD B., YOON S.-E., SUD A., MANOCHA D.: Interactive shadow generation in complex environments. In *SIGGRAPH 2003, Computer Graphics Proceedings* (2003), ACM Press / ACM SIGGRAPH, pp. 501–510. 4, 7, 9

[Gum99] GUMHOLD S.: Improved cut-border machine for triangle mesh compression. In *Erlangen Workshop '99 on Vision, Modeling and Visualization* (Nov. 1999), IEEE Signal Processing Society. 5

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. In *Eurographics State-of-the-Art Reports* (2003), pp. 1–20. 4

[ILGS03] ISENBURG M., LINDSTROM P., GUMHOLD S., SNOEYINK J.: Large mesh simplification using processing sequences. In *IEEE Visualization 2003* (2003), pp. 465–472. 3

[ISH03] ISARD M., SHAND M., HEIRICH A.: Distributed rendering of interactive soft shadows. *Parallel Computing 29*, 3 (2003), 322–323. 4

[KD03] KIRSCH F., DOELLNER J.: Real-time soft shadows using a single light sample. *Journal of WSCG 11*, 2 (2003), 255–262. 4

[KS01] KLEIN R., SCHILLING A.: Efficient multiresolution models. In *Festschrift zum 60. Geburtstag von Wolfgang Straßer* (2001), Schilling A., (Ed.), pp. 109–130. 2

[KSS98] KLEIN R., SCHILLING A., STRASSER W.: Illumination dependent refinement of multiresolution meshes. In *Proceedings of Computer Graphics International (CGI '98)* (Los Alamitos, CA, 1998), IEEE Computer Society Press, pp. 680–687. 3, 4

[Lev02] LEVENBERG J.: Fast view-dependent level-of-detail rendering using cached geometry. In *IEEE Visualization* (2002), pp. 259–266. 2

[Lin02] LINDSTROM P.: Out-of-core construction and visualization of multiresolution surfaces. In *ACM SIGGRAPH 2003 Symposium on Interactive 3D Graphics* (2002), pp. 93–102, 239. 3

[LS01] LINDSTORM P., SILVA C. T.: A memory insensitive technique for large model simplification. In *IEEE Visualization* (2001), IEEE. 3

[MHE*03] MCGUIRE M., HUGHES J. F., EGAN K. T., KILGARD M. J., EVERITT C.: Fast, practical and robust shadows. http://developer.nvidia.com/object/fast_shadow_volumes.html, 2003. 3

[Pom00] POMERANZ A. A.: Roam using surface triangle clusters (rustic). Master's thesis, University of California at Davis, 2000. 2

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 343–352. 2, 3, 5, 8

[Ros99] ROSSIGNAC J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (/1999), 47–61. 5

[SCH*01] SHOU L., CHIONH J., HUANG Z., RUAN R., TAN K. L.: Walking trough a very large virtual environment in real-time. In *Proceedings International Conference on Very Large Data Bases* (2001), pp. 401–410. 2

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proceedings of ACM SIGGRAPH 2002* (July 2002), Hughes J., (Ed.), ACM Press/ ACM SIGGRAPH. 3, 7

[SG01] SHAFFER E., GARLAND M.: Efficient adaptive simplification of massive meshes. In *IEEE Visualization* (2001), IEEE. 3

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proceedings of the 13th Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 87–98. 3

[TCRS00] TARINI M., CIGNONI P., ROCCHINI C., SCOPIGNO R.: Real time, accurate, multifeatured rendering of bump mapped surfaces. *Computer Graphics Forum (Eurographics 2000) 19*, 3 (2000). 3

[VM02] VARADHAN G., MANOCHA D.: Out-of-core rendering of massive geometric environments. In *IEEE Visualization 2002* (2002). 2

[WH03] WYMAN C., HANSEN C.: Penumbra maps: Approximate soft shadows in real-time. In *Proceedings of the 2003 Eurographics Symposium on Rendering* (2003), pp. 202–207. 4

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics archive 12*, 3 (1978), 270–274. 3

[WLC*03] WILLIAMS N., LUEBKE D., COHEN J. D., KELLEY M., SCHUBERT B.: Perceptually guided simplification of lit, textured meshes. In *Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), ACM Press, pp. 113–121. 3