Exploiting Multiresolution Models to Accelerate Ray Tracing

G. Papaioannou, E.A. Karabassi, H. Fretzagias, T. Theoharis

Department of Informatics
University of Athens

Abstract
In this paper, it is shown how multiresolution models can be exploited in order to improve the efficiency of the ray tracing process without significant image deterioration. To this effect a set of criteria are established which determine the level of detail (LOD) model that should be used for each ray-object intersection. These criteria include the distance from the observer and the amount of distortion to which a ray has been subjected before hitting the object. The resulting images are indistinguishable to the naked eye from those obtained using the maximum resolution models but require significantly less time to compute. Our method can be used in conjunction with previous ray tracing acceleration methods.

keywords: multiresolution models, LOD, ray tracing.

1. Introduction

Ever since the introduction of ray tracing [Whit80] there have been constant attempts to reduce its overwhelming time complexity, which is mainly due to the large number of ray-object intersection tests. Such techniques have evolved in two directions: the parallelisation of the algorithm [Kim96, Notk97, Park99] and the introduction of clever ways to reduce the number of intersection tests, such as the use of bounding volumes or space-subdivision [e.g., Glass84, Wegh84, Form95].

Today computer modelling advances and 3D scanning techniques allow the creation of extremely complex models, with an obvious impact in the computational cost of most rendering algorithms, including ray tracing. The presence of a highly detailed mesh in a scene dramatically increases the number of ray-triangle intersection tests performed; however this is not always reflected by the final result. A detailed mesh could often be replaced by a much simpler one, without significant deterioration in image quality, especially if it were indirectly visible through refraction or reflection. Consider for example a mesh with tens of thousands of triangles which is behind a highly distorting transmissive medium, e.g. a bumpy semitransparent object. If the occluded parts of the mesh were replaced by a similar mesh with only half as many triangles, or even less, the difference would hardly be visible, if at all.

James Clark [Clark76] has described the benefits of representing objects within a scene at several levels of detail, as early as 1976, and this approach is gaining much ground in conventional rendering methods. Multiresolution models [Heck94, Eck95, Ronf96, Heck97] aim to improve rendering performance while maintaining as much as possible the image quality. This is achieved by selecting the appropriate resolution of a given object dynamically, based on the perceptual importance of the object in the scene [He96]. In conventional renderers, this perceptual importance can be defined by various parameters such as the

distance of the object from the viewpoint or the total screen area occupied by the projection of the object.

In this work we bring multiresolution modelling to ray tracing, with the aim of reducing the amount of time spent on each ray-object intersection without significant image deterioration. Our method exploits the multiresolutional representation of rendered objects at all levels of the ray tree in order to speed up the ray-object intersection time. We introduce a set of criteria for selecting the object level of detail (LOD), based on the apparent distortion of the models as seen from the viewpoint. The technique presented here can be added onto previous acceleration techniques for further performance gain. Although the tests we performed were on polygonal meshes our method can be applied to any kind of multiresolutional models such as voxelised data or parametric surfaces [refs].

2. Algorithm Overview

In the following text we will use the notation $(\bar{P},\vec{D})$ to refer to a ray cast from point $\bar{P}$ that travels along a unit direction vector $\vec{D}$. We will represent the entire ray tracing process by a ray-tree, whose *i*-th level nodes correspond to the rays cast at the *i*-th level of recursion. Level 0 rays are assumed to be all rays cast from the viewpoint towards the scene. In this work, we regard the LOD of a model from the computational point of view and not the model aesthetics. In order to abstract the number of different resolutions a given object may be represented by, the LOD is measured in a normalised scale: LOD(*Object*)$\in$[0,1]. For polygonal models, a 0.5 model quality means that the desired resolution mesh must have at least half the polygons of the original high quality object. If $L_k(Object)$, $k = 0,\ldots lvl_{\max}$ are the multiple resolutions of *Object*, $L_0(Object)$ being the highest resolution representation, the appropriate model *k* is selected as:

$$k = lvl_{\max} - \lceil \mathrm{LOD}(Object) \cdot lvl_{\max} \rceil$$

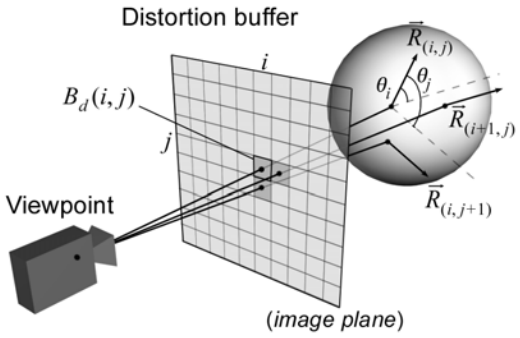
2.1. Level of Detail Selection Criteria

The overall shape of an object is distorted when secondary rays from the viewpoint reach the object surface in the following cases:

- When the rays have been first reflected on a curved surface.
- When the rays have travelled through a curved refractive medium.
- When the rays have been previously dispersed because of a highly bumpy interface.

In other words, the viewer's impression of an object A, which is visible via reflection on or refraction through an object B, is distorted when the ray disparity caused by the interfering object B is high. The term *ray disparity* is used to describe the angular divergence of an outgoing ray from object B when the incoming ray is slightly displaced. By measuring the ray disparity caused by the interfering objects, we can establish a criterion for the determination of the desired LOD of object A.

Figure 1. Ray disparity calculation. The direction of each first level ray reflected or refracted is stored in the distortion buffer.



In case of multiple interfering objects the deterioration of the apparent shape is cumulative, therefore we should take into account the total ray disparity. However, in typical scenes the percentage of rays that hit reflective or refractive objects at the 0^{th} level of recursion is significantly larger than the rest. Additionally, at this level the ray disparity can be easily calculated with a very small cost, as we will show in Section 2.2. As a result, we explicitly calculate the disparity taking into account only the initial level hits and avoid similar calculations at subsequent levels.

Apart from the initial ray disparity, the selected LOD is also affected by the recursion depth and the ray attenuation. The later is estimated based on the reflectivity factor and the surface opacity of the interfering media. The LOD selection functions with respect to all the above attributes are discussed in Section 3.

We should note at this point that an idea similar to the ray disparity, the *ray differential* is introduced by Igehy in [Igeh99] to efficiently filter texture maps in ray tracing, based on the divergence of neighbouring rays. Igehy hints that such divergences might be useful to the selection of an object's LOD, although he does not pursue any further this topic.

2.2. The modified ray tracing algorithm

The ray tracing recursion is broken down in two passes. In the first pass, all first level rays are traced and the intersections with the scene models are calculated using the highest level of detail. In case additional perceptual importance criteria are implemented, such as the distance of the object bounding box from the viewpoint, the LOD resulting from these criteria of the object is used as the high resolution model at this stage.

For every successful hit of a ray emanating from the image pixels (i, j), if the intersected object is reflective or transmissive the corresponding intersection point $I_{(i,j)}$ and the directions of reflection and refraction are stored in a *distortion-buffer* B_d of equal size with the rendered image. The (i, j) cell of this buffer holds the reflected and transmitted rays $(\bar{I}_{(i,j)}, \vec{R}_{(i,j)})$ and $(\bar{I}_{(i,j)}, \vec{T}_{(i,j)})$, respectively. Hence, the distortion-buffer stores the root nodes of the level 1 ray sub-trees (*Fig.* 1). If the object hit by the ray in the (i, j) cell of the buffer is opaque, the corresponding sub-tree is empty and the cell is marked appropriately. For the

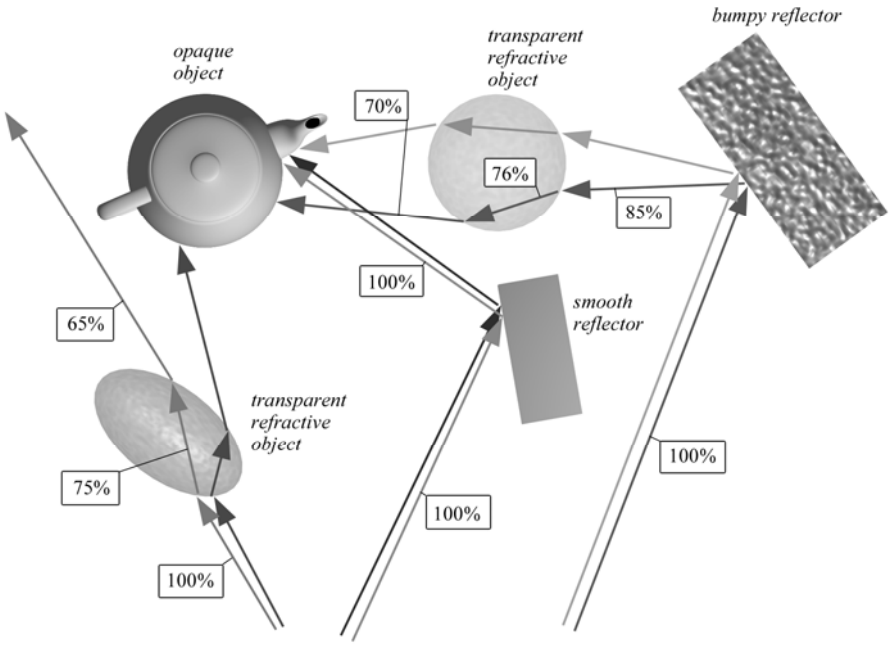


Figure 2. Model LOD degradation. Rays refracted or reflected on irregular surfaces use lower model resolutions. The LOD is further decreased due to ray attenuation. Note that the rays that hit the smooth reflector exhibit no ray disparity and therefore the desired LOD is not decreased.

calculation of the reflection and transmission vectors, we take into account both the model geometry and the locally varying textural attributes (bump mapping).

After the completion of the 0^{th} level ray-casting, the ray disparity is calculated using the ray directions stored in the distortion-buffer. The disparity $d(i, j)$ corresponding to pixel (i, j) reflects the mean planar angle between the outgoing ray at (i, j) and its neighbouring pixels, thus, having an intuitive meaning. Unless the image resolution is too poor, two neighbours, one horizontal and one vertical, are adequate. The desired LOD, which depends on the disparity at level 1, is stored in each outgoing ray and is progressively degraded in successively spawned rays. As the intersection of the incident ray and a model at point $\overline{I}_{(i,j)}$ spawns two new rays in the general case, separate disparity values $d_r(i, j)$, $d_t(i, j)$ are estimated for the reflected and refracted rays respectively:

$$d_r(i, j) = \frac{1}{2\pi}\left[\arccos(\vec{R}_{(i,j)} \cdot \vec{R}_{(i+1,j)}) + \arccos(\vec{R}_{(i,j)} \cdot \vec{R}_{(i,j+1)})\right]$$

$$d_t(i, j) = \frac{1}{2\pi}\left[\arccos(\vec{T}_{(i,j)} \cdot \vec{T}_{(i+1,j)}) + \arccos(\vec{T}_{(i,j)} \cdot \vec{T}_{(i,j+1)})\right]$$

where $\arccos(a) \in [0, \pi]$ and $d_r(i, j), d_t(i, j) \in [0,1]$.

An important observation is that the above expressions are image resolution dependent, because the resulting angles are not normalised by the pixel dimensions. This is a desirable property of the measured ray disparity as the models' LOD should also depend on the image

size. Note that in the calculation of $d_r(i, j)$ and $d_t(i, j)$ the fact that neighbouring hits of level 0 rays may belong to different scene objects is not taken into account. The image distortion implications of using different LODs at neighbouring pixels are only visible when the corresponding rays hit the same surface. Disparity and therefore LOD discontinuities at object silhouettes or occlusion transitions have no tractable effect on the image. Nevertheless, due to the fact that we only calculate the disparity for the primary hits only, it is a fair assumption that each object covers enough rendered image area for a large number of samples to hit the same object. In the opposite case where the footprint of an object on the image is small (for instance, less than 25 pixels), the shape of the reflected or refracted scene parts can not be recognised anyway.

In the second pass, the algorithm traverses the level 1 ray sub-trees and the ray-object intersection tests are performed with degraded versions of the models that were used in the first pass. The choice of the LOD for each model at depth $i > 0$ depends on the disparity estimated at level 1 and is further decreased due to the remaining selection criteria proposed in 2.1 (*Fig.* 2). The shadow feeler rays cast at any recursion depth use the same LOD as the incident ray.

3. The LOD selection functions

The disparity values can be directly used to associate a LOD with every level 1 outgoing ray. If $\mathrm{LOD}^{(0)}$ is the initially selected level of detail value of a ray originating from the viewpoint, the level 1 outgoing ray's LOD, $\mathrm{LOD}^{(1)}$, can be derived from:

$$\mathrm{LOD}^{(1)} = \mathrm{LOD}^{(0)} \cdot (1 - d(i, j))$$

In this case the degradation of the model's LOD is proportional to the local deviation of the outgoing rays. In practice, slight alterations in the ray's deflection direction have a dramatic impact in the apparent distortion of the reflected/refracted objects. To compensate for this non-linearity of the effect, we introduce a *distortion factor* $f > 1$:

$$\mathrm{LOD}^{(1)} = \mathrm{LOD}^{(0)} \cdot (1 - d(i, j))^f$$

Typical experimental values for f that select efficient models without causing any visible distortion are in the range [10,24].

During subsequent ray tracing recursions, the LOD is further decreased due to the ray's attenuation on non-perfect reflectors or through semitransparent materials. In fact, the LOD for rays of level 2 or greater, is only decreased when the ray disparity $d(i, j)$ measured in the first stage is significant, that is, greater than a predefined threshold $thres_{\mathrm{LOD}}$. Otherwise, we assume that the surface hit by the ray is smooth enough to prevent any visual distortion of the reflected/refracted objects. The ray attenuation depends on the reflectivity k_r of the model surface, the transparency k_t of the refractive medium and the material absorption $k_{abs}(l)$ with respect to the distance l travelled by the ray through the object. All the above coefficients are summarised in the attenuation factor k:

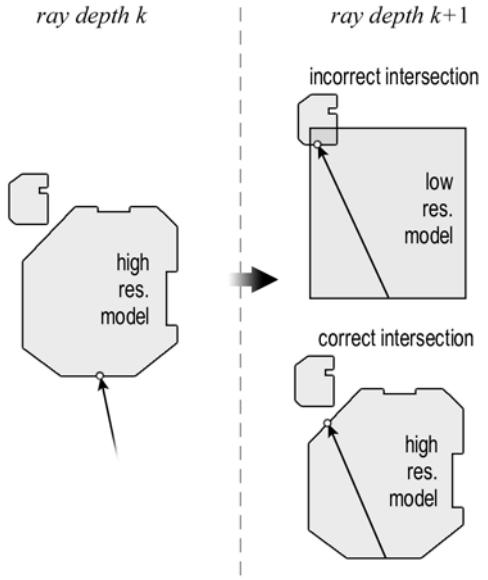


Figure 3. Handling inconsistencies when a ray's LOD is decreased between entering and exiting an object.

$$k^{(m)} = \begin{cases} k^{(m-1)} \cdot k_r^{(m)}, & \text{in case of reflection} \\ k^{(m-1)} \cdot k_t^{(m)} \cdot (1 - k_{abs}^{(m)}(l)), & \text{in case of transmition} \end{cases}$$

where *m* is the recursion level. *k* is stored in the ray's structure and is always calculated as an integral part of the basic ray tracer.

When $d(i,j) > thres_{\text{LOD}}$, the LOD is decreased proportionally to *k*, according to a *degradation rate* a_{LOD}:

$$\text{LOD}^{(m)} = \lambda^{(m)} \cdot \text{LOD}^{(m-1)}$$

$$\lambda^{(m)} = \begin{cases} 1, & d(i,j) > thres_{\text{LOD}} \\ 1 - a_{\text{LOD}} + a_{\text{LOD}} \cdot k^{(m)}, & \text{otherwise} \end{cases}$$

The constant $a_{\text{LOD}} \in [0,1]$ controls the rate of LOD decrease. For a_{LOD} =0, $\text{LOD}^{(m)} = \text{LOD}^{(m-1)}$, i.e. the ray attenuation is ignored and only the initial LOD reduction due to ray disparity is taken into account for the entire ray tree. Typical values for the degradation speed are in the range [0.4, 0.6].

A special case arises in the following circumstances: Suppose a ray enters a model using a $\text{LOD}^{(m)}$ and exits using $\text{LOD}^{(m+n)} \neq \text{LOD}^{(m)}$, after encountering *n*-1 more surfaces. Using different LOD when entering and exiting the same object can cause unexpected problems, e.g. in case of objects in close contact, as can be seen in *Fig.* 3. To handle these situations, each ray keeps track of the objects it has entered. Each time the ray enters a new object it adds its identifier to a list, along with the current LOD, $\text{LOD}^{(m)}$. When the ray leaves the same object, after *n* recursions, it disregards the current LOD, $\text{LOD}^{(m+n)}$, and uses, for this intersection test only, the previously stored $\text{LOD}^{(m)}$. After that, it removes the entry from the list.

4. Implementation

Although the use of multiresolution models drastically decreases the rendering time, some of the usual ray tracing speed-ups were adopted in our implementation. These include oriented bounding boxes, rejection of low strength rays and parallelisation.

In addition to these general speed-up techniques, we have implemented a selective tracing algorithm, which eliminates branches of the ray tree with relatively low strength after a predefined ray tracing level, provided that the LOD used is less than 100%. In this case, for each ray of level 2 or higher, if both reflected $(\bar{P},\vec{R})$ and transmitted rays $(\bar{P},\vec{T})$ are created at an intersection point $\bar{P}$, we compare the strength values of the two rays. If the strength ratio exceeds 200%, we discard the lower strength ray as it will have a rather insignificant contribution.

Ray tracing is an inherently parallel algorithm. Since we aim at performance we have considered parallelism in the implementation of the LOD algorithm and the result is a general *client/server* platform suitable for both *loosely* and *closely coupled* networks of parallel processors. The client process broadcasts the scene description to all available servers where the actual ray tracing is performed. Load balancing is dynamically performed in image space, each ray-tracing server rendering a span of image pixels.

The servers perform the 0^{th} level ray casting according to the initial partitioning and the resulting distortion-buffer is then sequentially processed to determine the LOD decrease for each pixel. Finally, the level 1 ray sub-trees are processed in parallel. All ray-tracing servers start rendering according to the initial partitioning. However a server thread that finishes its task, takes half the load of the most heavily loaded process. This dynamic partitioning resulted to a practical speedup of about 0.9·(number of processors).

5. Results

We have run several tests on an SGI Origin 2000 server with 512MB memory and 4 processors, each running at 180MHz, producing fully antialiased 512×512 pixel images (2×2 pixel supersampling). We used a variety of scenes, ranging from simple ones, without many reflective or transparent media, to very complex ones (see figures 4 and 5). During the tests, we used triangular meshes with up to 8 different levels of detail. The multiple LODs of the scene objects were generated externally with the help of commercial software.

One test case is presented in *Fig.* 4 where the rendered scene consists of a wooden chessboard with glass tiles and pawns. When we generated the image using only the high-resolution models we measured about $73\cdot10^9$ ray-triangle intersections. When multiresolution models were exploited for all scene models, we reduced the intersection number to approx. $49\cdot10^9$, resulting to a rendering time of 380min instead of 588min for the high resolution models (35% speedup). As can be observed from the distortion map components displayed in the same figure, the LOD is decreased on highly curved surfaces. By visually comparing the high-resolution and multiresolution images, it is not possible to spot any difference. We had to subtract the images and enhance the result in order to pinpoint any colour alterations (*Fig.* 4*e*).

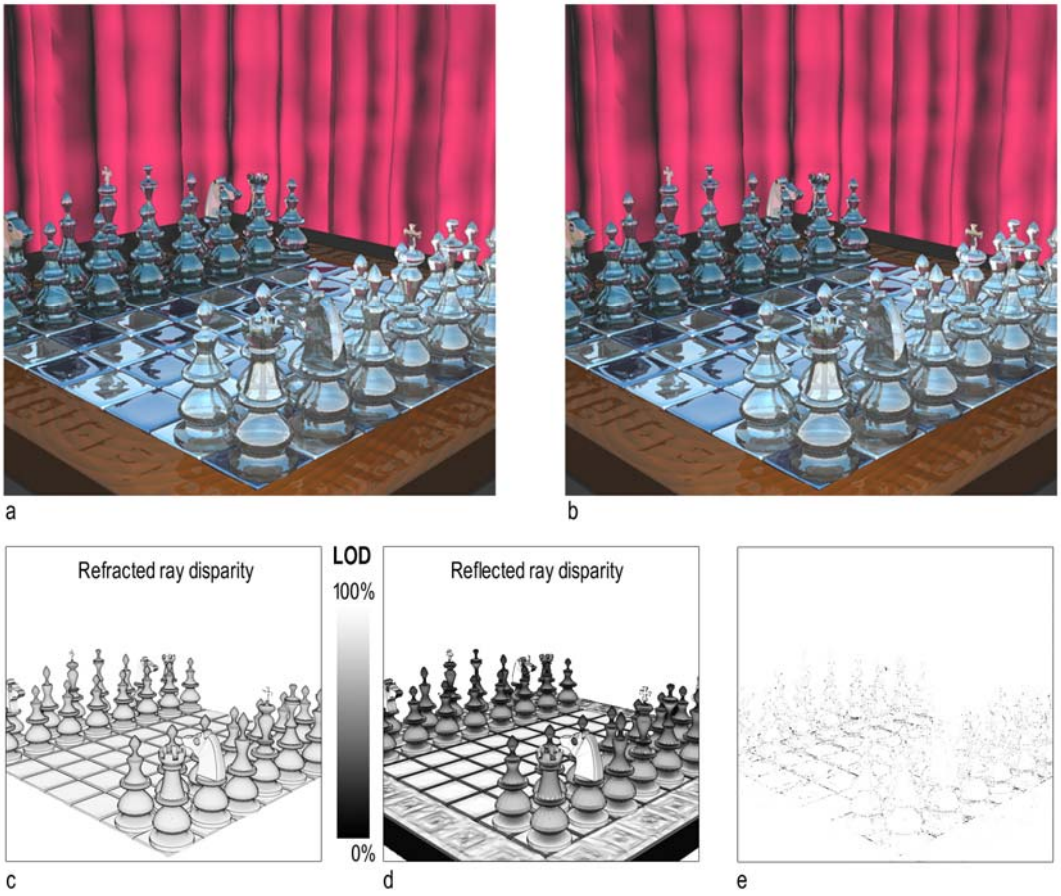


Figure 4. A ray-traced chessboard. (a) Without multiresolution models. (b) With multiresolution model support. (c) The refraction component of the distortion map. (d) The reflection component of the distortion map. (e) The image difference between image (a) and (b). The darkest pixels correspond to a maximum difference of 11%.

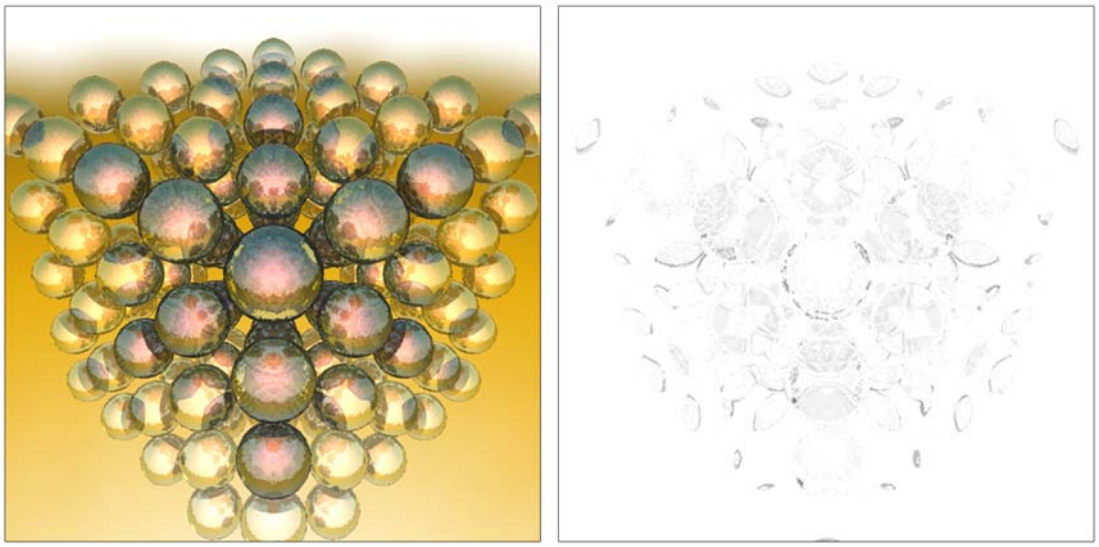


Figure 5. A good example of ray tracing time reduction using multiresolution models (left). The difference between the image rendered with high detail models and the one created using multiresolution models is displayed on the right. The maximum pixel difference measured was 16%.

Figure 6. Rendering the reflection of multiresolution models in a flat mirror (left). For rays that hit the perfect flat mirror on the back of the cabinet the LOD is not decreased. The difference between high detail and multiresolution rendering is shown on the right. The darkest values correspond to a difference of 21%.

Finally, the test example of *Fig.* 6 demonstrates the behaviour of our method on non-distortional objects. The flat mirror on the back of the depicted cabinet reflects the glasses in front of it (1st level rays) but since it is a flat surface, the reflected objects are not distorted and therefore the LOD of the reflected rays is not decreased. As too many rays hit the mirror at level 0 of the ray tree, we only got a speedup of 4% in this case.

References

[Whit80] T. Whitted, *An Improved Illumination Model for Shaded Displays*, Communications of the ACM **23**(1980), no. 6, 343-349.

[Park99] S. Parker, M. Parker, Y. Livnat, P.P. Sloan, C. Hansen and P. Shirley, *Interactive ray tracing for volume visualization* IEEE Transactions on Visualization and Computer Graphics **5** (1999), no. 3, 251-267.

[Form95] A. Formella and C. Gill, *Ray tracing: A quantitative analysis and a new practical algorithm* The Visual Computer **11** (1995), no. 9, 465-476.

[Kim96] H. J. Kim and C.M. Kyung, *A new parallel ray-tracing system based on object decomposition* The Visual Computer **12** (1996), no. 5, 244-253.

[Notk97], I. Notkin and C. Gotsman, *Parallel progressive ray-tracing* Computer Graphics Forum **16** (1997), no. 1, 43-55.

[Wegh84] H. Weghorst, G. Hooper and D.P. Greenberg, *Improved computational methods for ray tracing* ACM Transactions on Graphics **3** (1984), no. 1, 52-69.

[Glas84] A. S. Glassner, *Space Subdivision for Fast Ray Tracing* IEEE Computer Graphics and Applications **4** (1984), no. 10, 15-22.

[Eck95] M. Eck, T. DeRose, T. Duchamp, H, Hoppe, M. Lounsberry and W. Stuetzle, Multiresolution Analysis of Arbitrary Meshes SIGGRAPH ’95 Conference Proceedings, Annual Conference Series, August 1995, 6-11.

[He96] T. He, L. Hong, A. Varshney and S, Wang, *Controlled topology simplification* IEEE Transactions on Visualization and Computer Graphics **2** (1996), no. 2, 171-184.

[Heck94]

[Heck97]

[Ronf96]

[Igeh99] H. Igehy, *Tracing Ray Differentials* SIGGRAPH ’99 Conference Proceedings, Annual Conference Series, August 1999, 179-183.