Radiosity Techniques for Virtual Reality - Faster Reconstruction and Support for Levels Of Detail

Tomas Möller
Clarus AB
Stora Badhusgatan 18-20
S-411 21 Gothenburg
Sweden
e-mail: tompa@clarus.se

Abstract

We present a method, aimed at Virtual Reality (VR) applications with illumination calculated by radiosity, that provides faster reconstruction of the radiosity function of non-uniform rational B-splines (NURBS) and ultimate support for levels of detail, LODs[1]. For each NURBS, an image, called an *illumination map*, which contains its illumination is computed. Since many target machines for VR-rendering has support for real-time texture mapping, the illumination map is texture mapped, using bilinear interpolation, onto the surface in order to reconstruct the radiosity function. We show that the polygon count, used when rendering, can be considerably reduced using our method. Also, by decoupling shading from geometry, our method supports LODs for VR in an ultimate way, since only one radiosity calculation is needed for every conceivable LOD. This also implies that the triangulation of surfaces could be altered, without recomputing the illumination, in order to trade-off real-time rendering performance by surface approximation of the NURBS. We have also implemented the method in a real world application with excellent results.

Keywords: radiosity, texture mapping, Virtual Reality.

1. Introduction

To enhance a three dimensional model with more photo realism, radiosity (introduced in [4]) can be used to calculate realistic illumination of the surfaces. The goal of using radiosity for Virtual Reality (VR) is to be able to render realistic images fast enough for animation and walkthroughs.

The radiosity solution of a model is independent of the viewer's position and view direction and it could therefore be calculated in advance and used later to reconstruct the illumination of the model in real-time. The standard procedure for reconstructing the radiosity function for VR, is to render all patches/elements separately with Gouraud shading. However, since adaptive sampling [2] must be used to generate an accurate radiosity solution, this results in rendering an enormous amount of triangles. Also, since the triangulation of a surface usually is used as a foundation for sampling the illumination, the triangulation could not be arbitrarily altered without recomputing the entire

[1] In this context we mean different triangulations of a surface when speaking of LODs.

radiosity solution, which means that the radiosity solution must be computed again if the triangulation of a surface is altered.

Different levels of details, LODs, for each object are usually supported by VR-systems. Using LODs means that surface approximation is more accurate close to the viewer and less accurate for distant objects. This makes for better rendering performance when the objects are far away from the viewer, since fewer triangles are used to render the surface, and better surface approximation when they are close, since more triangles are used. Naturally, LODs should be supported also when radiosity has been used to calculate the illumination of the model.

2. Faster Reconstruction of the Radiosity Function

The use of Gouraud shading when reconstructing the radiosity function is due to the fact that many systems support this in hardware. High-end graphics systems, workstations and game stations has begun to support real-time texture mapping in hardware and to obtain an improved reconstruction method, we utilize this feature.

To be able to use texture mapping for radiosity function reconstruction, the sample points of the surfaces must be carefully selected. The radiosity values of the sample points are sampled using ray tracing [7]. Since texture mapping an image onto a surface is performed in the surface's parametric uv-space, uniform sampling of the radiosity function in uv-space automatically implies a texture map. This means that the radiosity values must be sampled and stored in a texture map, which we call an *illumination map*, and the samples must be located uniformly in uv-space of the NURBS. Note that we select the sample points independently of the triangulation of the surface. The concept of uniform sampling on a surface is depicted in figure 1. In this figure an illuminated surface is shown to the left and the uniformly located sampling points are shown in the middle. We sample the radiosity values at these points and generate the illumination map to the right. Uniform sampling of Bézier patches was introduced in [5].

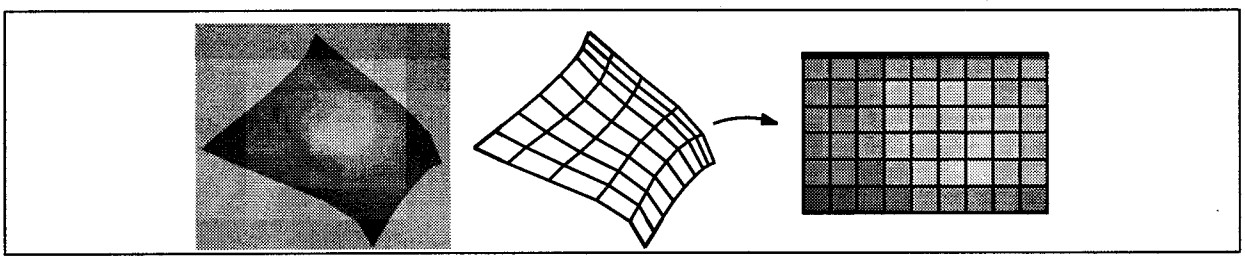

Figure 1: The concept of using uniform sampling in uv-space to generate a texture map is shown here. Note that the radiosities are sampled at the crossings of the lines.

If uniform sampling is used for a certain surface to generate an illumination map, the reconstruction of the radiosity function for that surface is straightforward. Simply texture map the generated illumination map onto the surface. These operations are shown in figure 2, where the illumination map to the left is texture mapped onto the triangulated surface in the middle and the result, the surface with reconstructed illumination, is shown to the right. Similar work has been presented in [6].

However, adaptive uniform sampling has to be used in order to sample high radiosity gradient regions more heavily. Our method does not immediately imply a texture map in this case, but this problem can be overcome by transforming the adaptive sampling bitmap to a common uniform bitmap. Making the adaptive sampling bitmap *balanced* [1], reduces this transform into a bilinear interpolation operation, which is applied recursively.

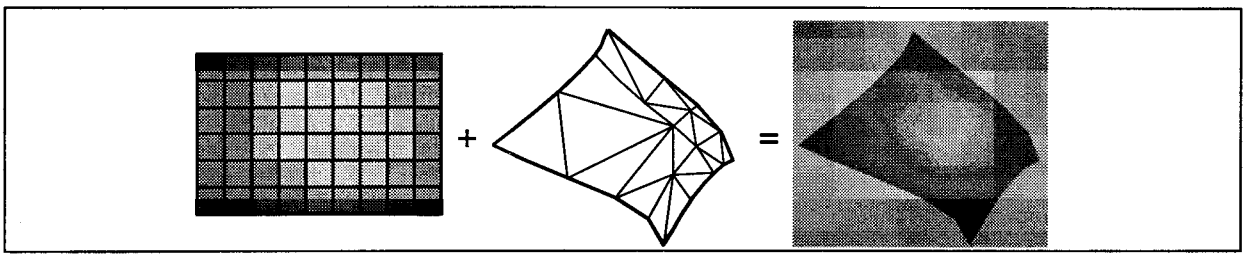

Figure 2: Radiosity function reconstruction using an illumination map.

An example of the adaptive to common bitmap transform is shown in figure 3. In this

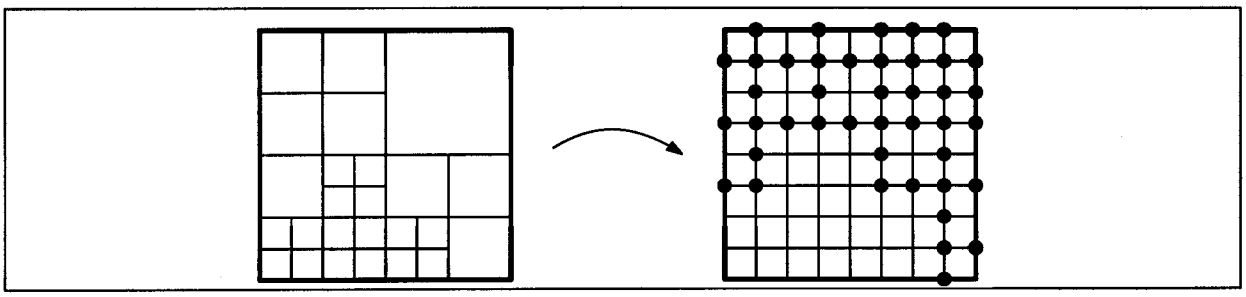

Figure 3: An adaptively subdivided bitmap (mesh) can be transformed – via bilinearly interpolating corner values – into a common (uniform) bitmap.

figure, the sample points are located at the line crossings and the dots in the destination bitmap are determined via bilinear interpolation. This technique is better than the anchoring method, described in [1], since all eight corner values are used when interpolating the middle point. A new value between two already computed values are computed by averaging them, and when this has been done the middle point can be weighted together from its eight neighbours. Figure 4 shows an example of this.

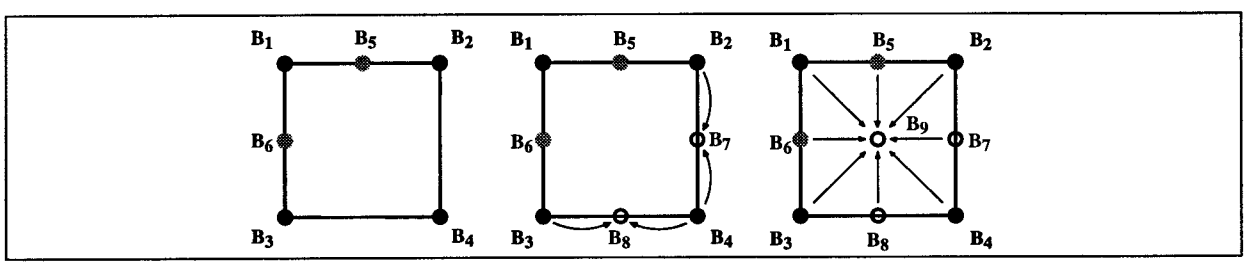

Figure 4: Assume that the left hand mesh is part of a larger mesh and that neighbouring parts have computed the radiosity values B_5 and B_6. When calculating the texture map for the mesh, several new values need to be weighted by old ones. The radiosity values B_7, B_8 and B_9 need to be computed for this example. B_7 and B_8 are weighted by B_2 and B_4 respectively B_3 and B_4. Finally the middle point is weighted by all eight radiosity values. Note that this mesh only contains sample points, it has nothing to do with the triangulation of the surface.

To replace blockiness with blurriness, i.e. replace sharp edges with soft edges, bilinear interpolation is used while texture mapping the illumination map onto the surfaces. Note that bilinear interpolation is rotation-invariant.

For surfaces with an initial texture map, a method based on [2] is used. The average reflectivity, $\rho_{average}$, of the initial texture map is used to generate an illumination map in

the same manner as for a non-textured surface. Then the illumination map and the initial texture map are scaled to the same size, using bilinear interpolation. For each pixel, the textured illumination map is computed by the following formula from [2].

$$B_{final}(u,v) = B_{illumination\ map}(u,v)\frac{\rho_{initial\ texture}(u,v)}{\rho_{average}} \tag{1}$$

$B_{final}(u,v)$ is the pixel value at (u,v) of the textured illumination map, $B_{illumination\ map}(u,v)$ is the pixel value from the computed illumination map and $\rho_{initial\ texture}(u,v)$ is the reflectivity for the current pixel. An example of handling initial textures within radiosity is depicted in figure 5, where the topmost image to the left contains the illumination of a surface and below is the initial texture map for that surface. These are scaled to the same size and multiplied, using (1), to form the illuminated texture map the right.

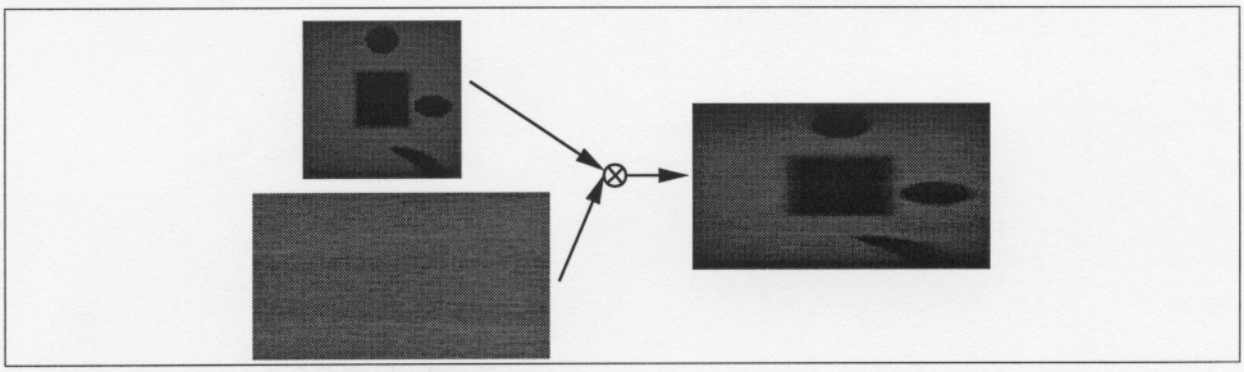

Figure 5: Handling of initial texture maps within radiosity.

3. LOD Support

To increase rendering preformance, most VR-systems support LODs. This means that surface approximation, which is controlled by the tesselation parameters of the NURBS, is more accurate when the surface is close to the viewer and coarser when the surface is farther away. For example, a distant surface that only covers 20 pixels on the screen can be approximated by a very small number (< 10) of triangles, without reducing the image quality too much. When the surface comes closer a finer triangulation is used to obtain a better approximation.

If a radiosity solution has been computed for rendering with Gouraud shading, the triangulation of the surfaces in the model cannot be altered without recomputing the radiosity solution.

Our method support LODs in an ultimate way, since only one radiosity calculation is needed for every conceivable LOD. This is true, since texture mapping is independent of the triangulation of the surface, that is, shading (the illumination map) is decoupled from geometry. An example of this is depicted in figure 6, where the illumination map at the top is texture mapped onto the different triangulations to the left. The textured surfaces are shown to the right.

A desired feature of a model for VR is to be able to fine-tune the real-time rendering performance by changing the surface approximation for the LODs. Our solution permits this kind of action even after the radiosity calculations. Fewer triangles in the model means more speed and less surface accuracy and vice versa.

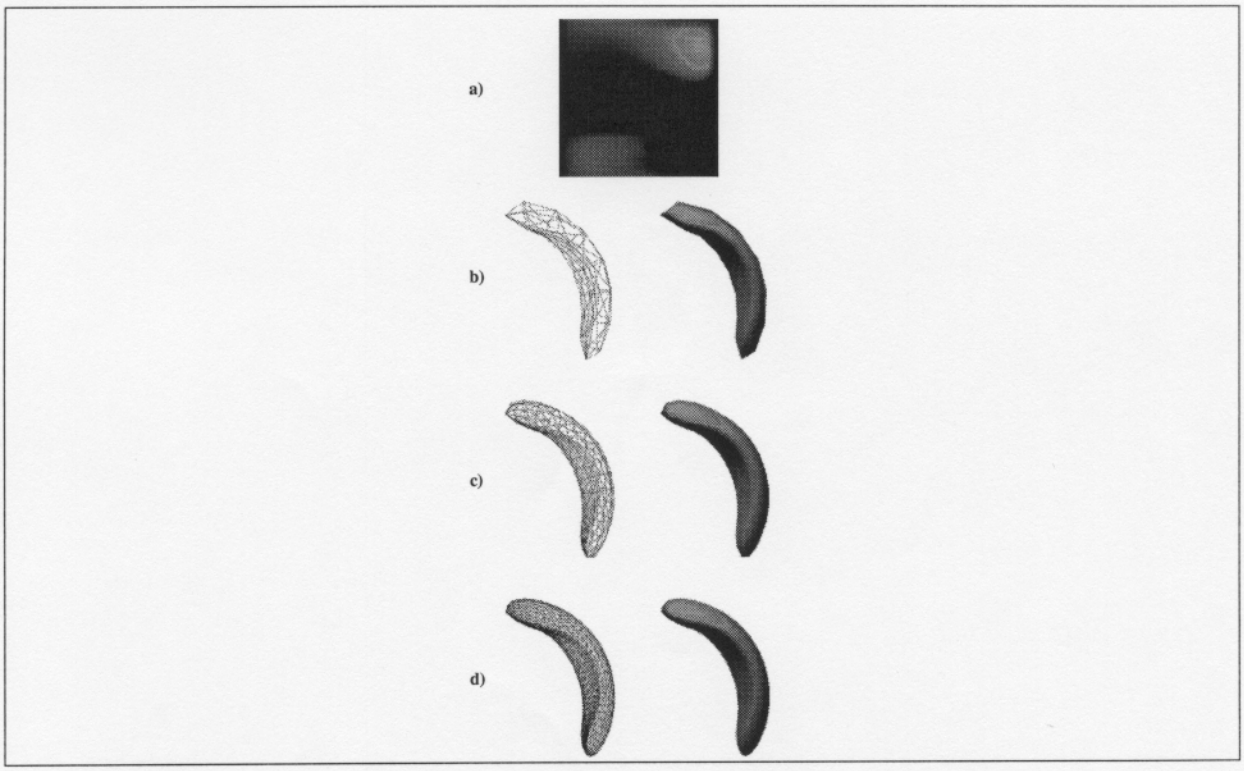

Figure 6: The same illumination map can be used with any conceivable triangulation of a surface, which means that LODs are supported in an efficient way.

4. Reducing the total texture size

When the radiosity calculations are finished, the result is a texture map for each surface. However, the sum, called S, of the amount of memory all texture maps occupy can exceed the amount of texture memory, called T. Now, to be able to render the scene efficiently S must be less or equal to T, $S \leq T$. If $S > T$ we must reduce S into S' so that $S' \leq T$. We propose a heuristic for solving this problem.

For each adaptively subdivided bitmap, find all the sample points at the deepest level of subdivision and compute the absolute value of the difference between these values and the values that would have been computed by bilinear interpolation (assuming we remove the sample points at the deepest level of subdivision). Call this sum D_i for surface i. Now, to reduce the total texture size, we remove the deepest level of subdivision, via low pass filtering, of the surface with the smallest D_i until $S' \leq T$. By doing this we, in some way, down sample the image that has the smallest amount of useful information on the deepest level of subdivision, which should be a good choice. Another approach could be taken if we have knowledge about the size of T (which is not always the case), before the radiosity calculations starts. In that case we could turn off further adaptive sampling when S has reached T, in order to restrict S.

5. Results

Three test scenes have been used to verify the function of the ideas in this paper. Those are depicted in figures 7,8 & 9 and are called *Fruit of the Room*, *The Monet Room* and *The Jussi Car*. All three scenes have illumination computed by radiosity and the solutions

have been reconstructed by the presented algorithm. The surfaces in the scenes are all NURBS and they have been triangulated before rendering.

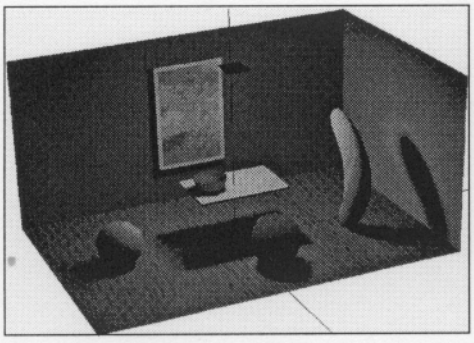

Figure 7: *Fruit of the Room.*

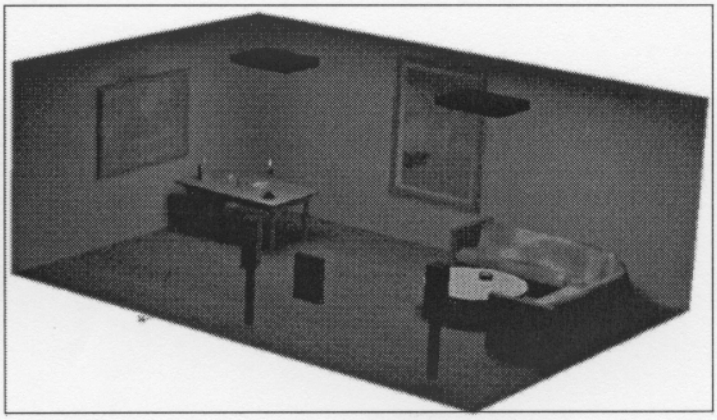

Figure 8: *The Monet Room.*

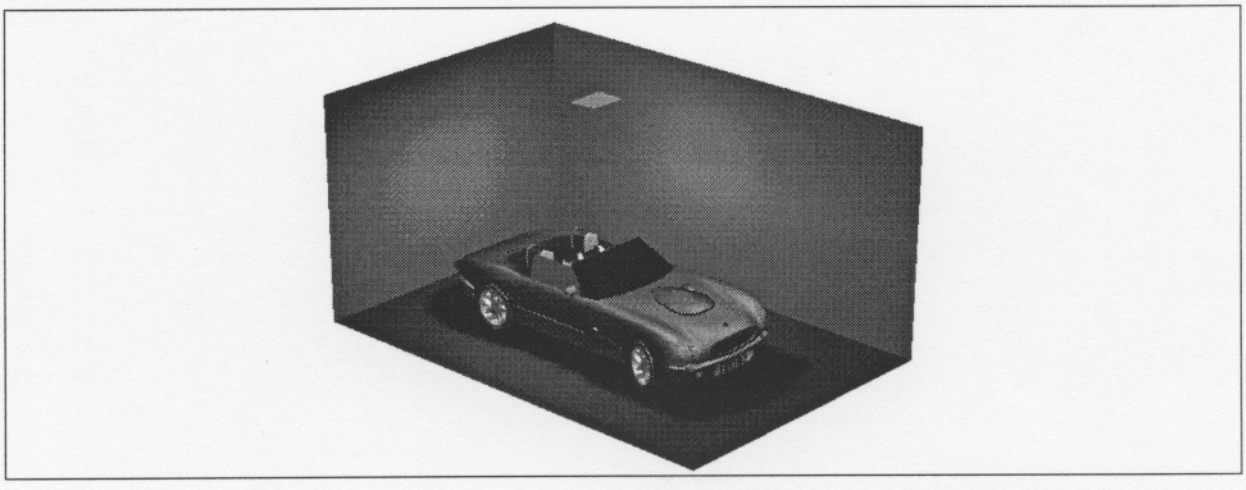

Figure 9: *Jussi Car.* Note that it is only the illumination of the floor and the walls that have been computed by radiosity. The car is only used to cast a shadow on the floor. Only one iteration step has been computed in the progressive refinement radiosity method [3].

In the table below, the traditional radiosity reconstruction method, that is using Gouraud shading, is compared to the texture mapping reconstruction method. The *Memory* columns show how much memory that was used for storing the vertices for the triangles plus the texture maps (if any) and the *Triangles* columns show the number of triangles needed to render the scene. The figures for *Jussi Car* only includes information for the floor and the walls, that is the illuminated surfaces. However, the figures in the parentheses includes the car as well.

	Gouraud		*Texture*	
	Triangles	*Memory*	*Triangles*	*Memory*
Fruit of the Room	11628	559.4 kB	2710	702.7 kB
The Monet Room	39908	872.4 kB	4018	3.5 MB
The Jussi Car	6618 (28084)	52.3 kB (288.2)	6 (21472)	46.2 kB (282.1)

Since Gouraud shading and texture mapping are equally fast on the target machine for rendering, this implies that the fewer the triangles the better the rendering performance. In the table we show that our method reduces the number of triangles considerably and therefore the real-time performance is improved with the texture map based reconstruction method. For the test scenes, the traditional Gouraud method consisted of between 4 to 100 times more triangles than the new method.

In many cases, for example rectangular surfaces like floors and walls, several hundred Gouraud shaded triangles are replaced by 2 texture mapped triangles. Since texture mapping is as fast as Gouraud shading, rendering speed is increased substantially. Since the number of initial sampling points of a surface is rather small (compared to after adaptive sampling), large surfaces like the walls need not occupy much memory. When adaptive sampling is activated the texture maps may grow and become large. However, as can be seen in the table, the texture maps of the floor and the walls in *Jussi Car* only occupy 46.2 kB, which is a rather small amount. We can use the presented method for triangles and quadrangles since they have a natural *uv*-space. Other polygons could be decomposed into triangles and quadrangles.

We have implemented the presented method in a real world application called *Clarus CAD Real-Time Link* (CRTL), which is a commercial program for CAD-translation and fine-tuning of VR-models. The surfaces that CRTL forwards to the radiosity program are all NURBS and for each NURBS, an illumination map is returned. We found that the texture map based method for radiosity reconstruction greatly improved rendering performance and supported LODs efficiently. The snapshots in figures 7, 8 & 9 have been taken directly from CRTL. All the radiosity solutions of the test scenes have been used for high-performance realistic Virtual Reality on an SGI ONYX. At the point of writing this, the presented techniques have been used successfully in several real projects, including for example walkthroughs of a robot industry and a Swedish project called Game-On-Demand.

6. Conclusions

More and more computers for real-time rendering have support for texture mapping in real-time. We have presented a method, aimed at VR-applications, for faster radiosity function reconstruction, that exploits this feature. The method also supports LODs in an ultimate way since only one radiosity solution needs to be computed. This solution can be used with any conceivable triangulations, i.e. LOD, of the NURBS. We have also shown, with test scenes, that the number of triangles needed to render the scene can be considerably reduced using the new method. Another advantage is that real-time rendering performance can be fine-tuned after the radiosity calculations in order achieve the desired frame rate in the VR-system. An important aspect of this paper is that the method has been implemented in a real world application and used in real projects.

A disadvantage is that the texture maps can occupy a lot of memory. This can be partially solved by using illumination maps for curved surfaces (the banana in figure 6 for example), that is surfaces that need a finer triangulation when they are closer to the viewer. Flat surfaces, like the floor and walls in the test scenes, could be rendered with the traditional Gouraud method, since they are rendered with the same triangulation on every LOD. This would decrease the amount of texture memory needed. Another

approach would be to reduce the total texture size with the proposed method.

References

[1] Daniel R. Baum, Stephen Mann, Kevin P. Smith and James M. Winget, *Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions*, Computer Graphics (SIGGRAPH'91 Proceedings) Vol. 25, No. 4, July 1991, pp. 51-60.

[2] Michael F. Cohen, Donald P. Greenberg, David S. Immel and Philip J. Brock. *An Efficient Radiosity Approach for Realistic Image Synthesis*, IEEE Computer Graphics and Applications, Vol. 6, No. 2, 1986, pp. 26-35.

[3] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace and Donald P. Greenberg, *A Progressive Refinement Approach to Fast Radiosity Image Generation*, Computer Graphics (SIGGRAPH'88 Proceedings) Vol. 22, No. 4, August 1988, pp. 75-84.

[4] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg and Bennet Battaile, *Modeling the Interaction of Light Between Diffuse Surfaces*, Computer Graphics (SIGGRAPH'84 Proceedings) Vol. 18, No.3, July 1984, pp. 213-222.

[5] A. Kok, C. Yilmaz and L. Bierens, *A Two-Pass Radiosity Method for Bézier Patches*, Eurographics 1989, pp. 115-124.

[6] K. Myszkowski and T. L. Kunii, *Texture Mapping as an Alternative for Meshing During Walkthrough Animation*, 5th Eurographics Workshop on Rendering, Darmstadt, Germany, 1994, pp. 374-388.

[7] John R. Wallace, Kells A. Elmquist and Eric A. Haines, *A Ray Tracing Algorithm for Progressive Radiosity*, Computer Graphics (SIGGRAPH'89 Proceedings) Vol. 23, No. 3, July 1989, pp. 315-324