Optimization Schemes for Ray Tracing

Jeppe Revall Frisvad, s991020@student.dtu.dk
Rasmus Revall Frisvad, s973867@student.dtu.dk
IMM, June 2004

DTU

INTRODUCTION

The current HW techniques for real-time rendering include more and more visual effects which we some years ago saw only in photo realistic rendering.

These visual effects include hard and soft shadows using shadow maps or stencilled shadow volumes, per pixel shading using physically based BRDF theory (eg. Cook-Torrance or Schlick), reflections and a single level of refraction by environment mapping, diffusely interreflected light usually stored in light texture maps. A large collection of these techniques is given in [1].

This poster is a teaser for those that have not considered how to optimize ray tracing using real-time rendering techniques.

METHODOLOGY

The first optimization is to do the first level of ray tracing in hardware. Make a simple vertex shader that scales the position of each vertex (in world coordinates) to the interval [0,1]. This can be accomplished using an AABB of the scene to be rendered.
First give all primitive objects a specific color id and draw the scene (see figure 1a), then draw the position of each fragment using Gouraud shading and the simple vertex shader mentioned above (see figure 1b). Those two images result in the first level of ray tracing (intersection point and id of the intersected triangle). From there on the ray tracing can carry on as usual. For further optimization the depth buffer could be used to specify the t value of the intersection point for each ray ($\mathbf{r} = \mathbf{o} + t\,\mathbf{d}$) that would eliminate the pass to make image (b).

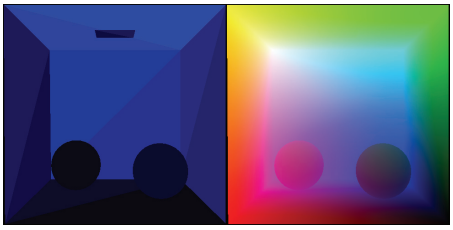

Figure 1. (a) *color ids for each primitive in the scene,* (b) *position of first intersection scaled to the interval [0,1].*

Next shadow volumes are an obvious replacement of shadow rays. For each light source render the shadow volumes to the stencil buffer as usual and use the result to decide whether light from a particular light source should contribute to the shade of each ray representing a pixel. Soft shadows are also possible using techniques such as penumbra wedges [2].

An opportunity is to do the BRDF calculations in a fragment shader. Both shadow volumes and BRDF calculations are, however, useful only for the first level of ray tracing.

The first reflection and refraction vectors for the recursion to level two of the ray tracing can be found in HW by a simple vertex shader. Environment mapping is not used since it is incorrect when the reflected objects are close to the reflector. Figure 2 is a real-time reference illustrating that shortcoming.

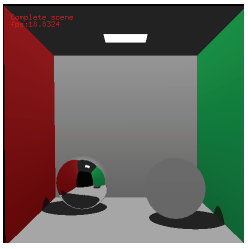

Figure 2. *Real-time rendering using stenciled shadow volumes and an environment cube map for each sphere. The left sphere is reflective while the right one is refractive. Notice that environment mapping provides only a single level of refraction and that the reflected shadow is incorrect due to the simplifying assumption that the environment is far away.*

RESULTS

The method presented results in the following speedup; rendering figure 3b takes 41% of the time we spend for the rendering of figure 3a. The slightly more complex scene of figure 4 takes app. 26% of the reference time. A network of solid angles between AABBs of the objects in the scene is the only spatial optimization scheme that we employ. Rendering times are given in the table below (1GHz Pentium 3 machine with a GeForce4).

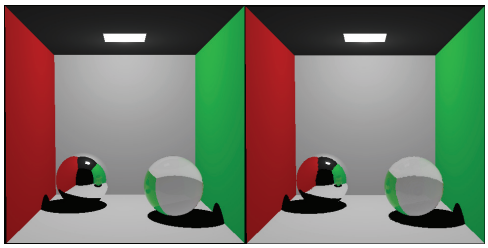

Figure 3. (a) *reference image using standard (one ray per pixel) ray tracing,* (b) *the same image found using HW optimizations.*

Method\Scene	Two Spheres	Orb on Pedestal
Standard	10.297 s	137.547 s
HW	4.218 s	36.125 s

The inaccuracy of the color buffer (8 bits per color band) leads to some aliasing artifacts close to the lines and highlights of reflections and refractions.

EXPANSIONS AND EXPERIMENTS

Using a fragment shader for the position image or a p-buffer with higher precision might eliminate the aliasing artifacts.

An experiment we did was to trace the rays after the first level by movement of the camera and rendering of a size 1x1 image for each ray. The bottleneck is, of course, that each ray must travel all the way down the graphics pipeline, which takes app. half a millisecond on a GeForce4 with a simple scene as the one above. This approach is hardly more efficient than a highly optimized spatial data structure, but on the other hand we need not reconstruct any data structure for each frame.

CONCLUSIONS

In this poster we employ a number of real-time techniques in order to speed up the process of ray tracing. Only techniques that do not simplify the illumination model beyond the simplifications of ray tracing are considered.

The entire first level of ray tracing emigrates to the GPU. Shadows are found using shadow volumes. An experiment has been to approximate the result of each reflected and refracted ray using images of size 1x1, which would move the entire process of ray tracing to the GPU. However, for that experiment to be successful we must await future graphics cards with an increased ability to render a large amount low resolution images.

References:
[1] Akenine-Möller, Tomas, and Eric Haines. *Real-Time Rendering*. A K Peters, second edition, June 2002.
[2] Assarsson, Ulf, and Tomas Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware.*ACM Transactions on Graphics*, 22(3), pp. 511-520, 2003.

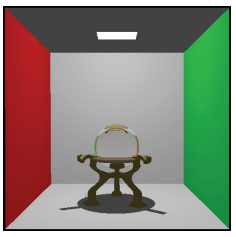

Figure 4. *A slightly more complex scene*