Two Optimization Methods for Raytracing

by

W. Stürzlinger and R. F. Tobler

Johannes Kepler University of Linz
Institute for Computer Science
Department for graphical and parallel Processing
Altenbergerstraße 69, A-4040 Linz, Austria, Europe

Tel.: +43(732)2468-9228, Fax : +43(732)2468-10
e-mail: wrzl@gup.uni-linz.ac.at

Two Optimization Methods for Raytracing

W. Stürzlinger and R. F. Tobler

Institute for Computer Science, Johannes Kepler University of Linz, Austria

Abstract

Raytracing is a method to produce photo-realistic pictures. For each pixel of an image one ray is shot to find the object, which determines the color of the pixel. Rays are also used to simulate shadows and reflections. Previously bounding volume hierarchies have been used to speed raytracing. A new method to speed the traversal of a bounding volume hierarchy is presented. Optimisation methods to find the intersection point between the ray and the scene have been introduced previously. A new algorithm based on cylinders is presented also.

1 Introduction

Raytracing is a method to produce photo-realistic pictures. Rays are used to determine visibility and to capture important illumination effects such as shadows and reflections. For a more complete introduction to raytracing see e.g. the book by Glassner [Gl89].

Intersecting a ray with a non-trivial object (e.g. free-form surfaces, sweeps, etc.) is a very time consuming operation. One way to speed this intersection process is to use simple objects as bounding volumes (BV's), e.g. axis-aligned boxes, spheres, slabs. As the BV encloses the object completely, we can first intersect the ray with the simpler BV and only if the ray intersects the BV proceed to compute the intersection with the original object. This proved to speed raytracing significantly. Another method to restrict the set of objects, which has to be intersected with a given ray, is to subdivide the space of the environment and determine beforehand, which objects lie in each of the subdivisions. When tracing a ray one has to determine, which subdivisions the ray traverses, and intersect the ray only with these objects, which are stored with the respective subdivision.

2 Optimized Bounding Volume Hierarchy Traversal

Intersecting a ray with thousands of BV's and the respective objects is still a timeconsuming operation. Most rays intersect only a small part of all BV's and objects. Clearly the performance can be improved by intersecting the ray with only the objects which lie approximately in the region of space the ray traverses. One way to quickly cull objects from consideration is a hierarchy of BV's - a bounding volume hierarchy (BVH).

2.1 Bounding Volume Hierarchy

Figure 1 describes how to construct the union of two BV's. When this process is applied to the total set of objects, we get a set of BV's numbering half the originial objects.

Reapplying the process recursively leads to a hierarchy of BV's and the top node of the tree describes the BV of the whole scene. Of course the effectiveness of the BVH depends on

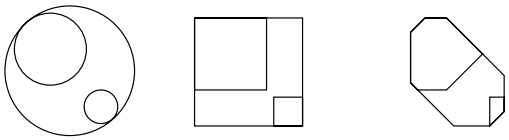

Figure 1: Combination of Bounding Volumes.

the order in which the objects are combined. If we build the BVH bottom-up by combining BV's randomly the resulting BVH will not show a good performance, as objects lying near to each other may end up in totally different parts of the BVH. Searching for the "nearest" BV is a time consuming process, therefore a good way is to build the BVH top-down. This is done by entering each object into the BVH so that the increase of the volume is minimized at each node. Another way to build the BVH is to utilized information supplied by the used in the modelling process.

In this paper it is assumed that either axis-aligned boxes or slabs are used to construct the BVH.

To intersect a ray with the scene stored in the BVH the ray is intersected with the topmost box. If an intersection is found, the children of the root node are tested for intersections and so on. When an intersection with a leaf of the BVH is found all objects inside this node are tested for intersection with the current ray.

2.2 Optimized Bounding Volume Hierarchy Traversal

The new approach exploits the special structure of a BVH with boxes and slabs more fully. When the union of e.g. two boxes is computed (see figure) a flag is stored, whether a side of the box conincides with the side of the box of the parent node (see figure 2).

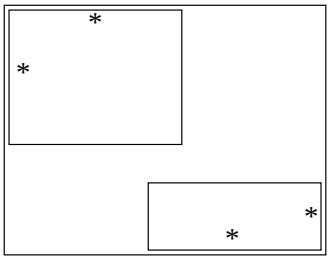

Figure 2: Conciding Box Sides (flagged).

When traversing the BVH there is no need to compute the intersection of the ray and a flagged side of the current box, as the intersection will have been computer previously

This number of BVH-box-intersection tests (more precisely the number of ray-plane-intersections) this improved algorithm executes is less than half the number of tests of the original algorithm. Therefore a speedup is to be expected.

3 Cylinder Cube

Previously a number of space subdivision methods to speed raytracing have been suggested. These methods include the regular grid, the octree, etc, which subdivide space into voxels. A "voxel walking" algorithm is then used to find the next voxel along the ray. When an intersection is detected in the current voxel the search is terminated, and the intersection is returned. Arvo and Kirk [AK87] proposed a method which subdivides the 5 dimensional ray space.

3.1 Cylinder Cube

Another way to exploit the coherence of scenes is to use the direction of the ray to eliminate objects from the intersection tests.

We start by partitioning each of the six sides of the bounding box of the total scene with a regular pattern of n^2 "gridels" (i.e. squares or rectangles, see figure 3).

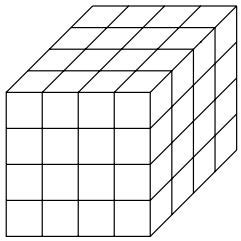

Figure 3: Cube with partitioned sides (gridels).

The interconnecting volume of two arbitrary gridels can be approximated easily by a cylinder. The two midpoints of the gridels are connected to form the central axis of the cylinder. Then this cylinder is enlarged until all of the eight corner points of the two gridels are included (see figure 4).

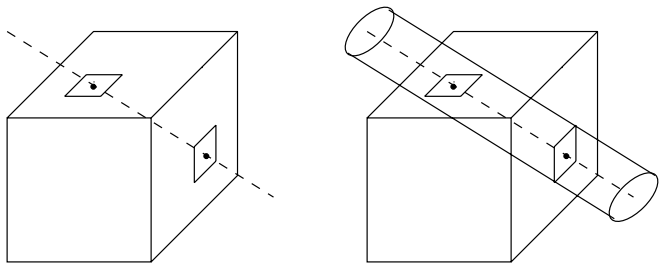

Figure 4: Construction of the cylinders.

The cylinder can be easiliy tested against a bounding sphere of an object, it is only neccessary to compute the distance of the center to the cylinder axis and compare this distance to the sum of the radii.

Thus we construct all cylinders of the cylinder cube and store for each of the $6n^4$ cylinders which objects lie inside or intersect the cylinder. Due to the symetry of a cylinder we need only store $3n^4$ cylinders.

To find which objects may potentially intersect a ray we intersect the ray with the bounding box of the scene. The two corresponding gridels (where the ray enters and leaves the bounding box) can be found easily by a modulo-operation. The index of the two gridels gives quickly the corresponding cylinder and the list of objects which lie inside.

4 Conclusion and Results

The optimized BVH traversal method speeds, as expected, the ray intersection test significantly. We measured the performance with a test scene showing a glass chessboard with glass figures (approx. 150 objects) and two diffuse chemical molecules consisting of approx 2800 and approx 27000 objects. The pictures were rendered on a SGI R3000 with 256 by 256 pixels.

Objects	Rays	Normal BVH Traversal	Optimized BVH Traversal
150	190000	185	143
2800	62000	91	74
27000	64000	184	158

As expected the optimized BVH traversal outperforms the old method consistently.

To test the cylinder cube optimisation method we compared it's results with the results for the regular grid method for the same scenes. The size of the grid was 4x4x4 and the cylinder cube was also subdivided into 6x4x4 cylinders.

Objects	Rays	Grid	Cylinder Cube
150	190000	186	199
2800	62000	65	202
27000	64000	114	-

For the last example we were not able to measure the time as the process ran out of virtual memory (48MB) after approx. 10 minutes. As one can see, normally the grid method is faster. One reason for this behaviour is that bounding spheres are not as tight as bounding boxes in general. This can also be observed for the figures for memory consumption, which were higher for the cylinder cube, due to the "looser" bounding volumes. The setup times (not shown) for the cylinder cube were also slower that the grid setup times due to the more complicated calculations.

References

[AK87] Arvo, J., Kirk, D., Fast Ray Tracing by Ray Classification, Computer Graphics (SIGGRAPH '87) **21**:4 (July 1987) 55–64

[Gl89] Glassner, A. S., An Introduction to Ray Tracing. Academic Press (1989).