Ray Tracing with Cones

John Amanatides

Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

SUMMARY

Introduction

Ray tracing is a very powerful yet simple approach to image synthesis [Whit80]. However, apart from the computationally expensive method of supersampling, no general method exists to remove artifacts created by undersampling. Furthermore, at present, there is no general procedure to decide what level of detail is sufficient in a texture map or in a procedural or hierarchical model of an object. This is primarily because individual rays are infinitesimally thick and thus we cannot exploit area-sampling techniques to avoid aliasing artifacts. This paper examines the use of rays that have a solid angle associated with them: cones.

Cones and Anti-Aliasing

When ray tracing, rays are shot from the eye into the world. They are constrained so.that they pass through the center of the pixels in the screen. Once they have left, any relationship with the pixel on which the results will be displayed is severed. This is because a ray is defined as a starting point and a direction which together form a line. This simple definition allows for straightforward and fast intersection calculations with various objects. The drawback, however, is that there is not enough information to anti-alias.

This paper extends the concept of a "ray" to that of a cone by including the angle of spread and virtual origin of the ray. With this information, the fraction of the intersection between a ray and an object can be calculated. This fractional coverage information is sufficient to perform simple anti-aliasing. Now, only one ray per pixel is sufficient regardless of scene complexity. The rays must simply be wide enough that they completely engulf the pixel on the screen as seen from the eye. Cones can also be used to anti-alias texture. At each intersection an estimate of the size of intersection can easily be calculated. This information can be used to average the texture map.

Fuzzy Shadows

Virtually all graphics systems model light as either a point source or as a direction from which parallel light beams emanate. Consequently, shadows cast by these light sources exhibit sharp boundaries. Cones allow us to extend our repertoire of light sources to include ones that cast fuzzy boundaries. For example, we can add spheres of varying radii as light sources. At each intersection, when a ray is sent to the light source to calculate the shadow, we broaden the ray to the size of the light source. By calculating how much of the light source intersects with the cone we can now generate fuzzy shadows.

Levels of Detail

A recurring problem with procedural and hierarchical models of objects is the level of detail to which they should be generated [see Clar76]. In ray tracing there is no good answer as there is no way of knowing how much of the screen an object will fill. Stochastic surfaces are a good example [Four82]. If we do not subdivide enough we will see the resulting polygons. If we subdivide too far, however, we will encounter two problems: First, we will waste computing resources and second, we will undersample. This is because further subdivision will introduce higher frequency components into the stochastic surface. We can use cones to advantage here also. By calculating the size of the intersection, we can decide what level of detail is sufficient. Thus two different views of the same object, one direct and one reflecting off another surface, can both be rendered correctly.

Specular Approximations

In his classic paper [Whit80], Turner Whitted raised the issue of generating specular highlights using ray tracing. His approach, however, had aliasing problems and fired off many rays at each intersection point, and was thus abandoned. We produce similar results by simply broadening the reflected ray. When rays are broadened, surfaces become less glossy. This results in reflections that are less pronounced.

Reducing Intersection Calculations

The cone approach provides a basis for reducing the number of intersection calculations required for ray tracing. By recursively firing wide cones at the screen, we can perform a Warnock style culling process. This can significantly reduce the number of intersection calculations required at each pixel.

References

Clar76 James H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms", Communications of the ACM, 19(10), October 1976, pp. 547-554.

Four82 Fournier, A., Fussell, D., and Carpenter, L., "Computer Rendering of Stochastic Models", Communications of the ACM, 25(6), June 1982, pp. 371-384.

Whit80 Turner Whitted, "An Improved Illumination Model for Shaded Display", Communications of the ACM, 23(6), June 1980, pp. 343-349.