Rendering Volumes With Microvoxels

Andrew Clinton∗  Side Effects Software Inc.
Mark Elendt†  Side Effects Software Inc.

1 Introduction

The REYES [Cook et al. 1987] rendering architecture is a well known divide and conquer rendering algorithm. The algorithm subdivides complex primitives into micropolygons which are then shaded and sampled to produce final pixel colors. The most important advantage of the REYES algorithm over other rendering approaches such as raytracing is that shading quality is fully decoupled from image sampling - making it possible to adjust image sampling quality without affecting shading.

Volumetric effects are commonly produced in REYES rendering systems using custom raymarching code to step through the volume and accumulate shading samples. The problem with this approach is that integrated features of the rendering system (such as motion blur, depth of field, and compositing) are difficult or impossible to use in conjunction with volume primitives.

We have extended the REYES algorithm to fully support volume primitives. Just like surfaces, volume primitives are diced into microvoxels - an entity analogous to a micropolygon, but subtending a 3D volume of space. Microvoxels are then shaded and sampled along with micropolygons. Our volume primitives are also hybrid raytracing primitives, making it possible to render secondary effects using raytracing.

2 Shading

Volume primitives are subdivided using a flexible dicing scheme that permits shading quality adjustment, view-dependent tessellation, and adaptive subdivision. The result of the dicing algorithm is a sequence of microvoxels filling the volume primitive. Commonly used features of surface rendering such as opacity culling work equally well with volumes, as the dicing procedure inserts both surfaces and volumes into a depth-ordered priority queue.

Given a sequence of voxels, we run the full REYES shading pipeline on the microvoxel corner points. This includes displacement shaders, which can move around the shading points in an arbitrary fashion before passing them to the surface shader. The surface shader is then responsible for computing the color and opacity for each shading point. An additional global variable $dPdz$ indicates the depth of the volumetric region that should be composited (this variable is set to 0 when shading surfaces).

One powerful feature of the REYES architecture is the ability to bind arbitrary geometric information to shader variables. In our implementation, user-defined volumetric fields are part of our geometry specification, making it possible to pass any desired value by name to the surface shader. Well-known bindings include density and temperature for shading of fluid simulations, and velocity for motion blur.

3 Sampling

Once microvoxels are shaded, we run a decoupled pixel sampling algorithm to generate sample lists. The sampling algorithm runs a fast SIMD raymarcher to generate hit lists for each subpixel sample, interpolating the shaded results for each hit. The same sampling parameters are used for volumes and surfaces, allowing several important features of surface rendering to work equally well with volumes.

Interleaved Transparency: Sample lists maintain z-order, so surfaces and volumes can be arbitrarily interleaved with correct compositing.

Motion Blur, Depth Of Field: Subpixel samples have an associated time and depth of field sample. Voxel motion is handled by moving the voxel vertices to the time for each subpixel sample, and then performing intersection tests against the transformed voxel edges. Depth of field simply changes the raymarch origin and direction.

Deep Image Export: Since volumetric sample lists are visible to the renderer, it is easy to export them to deep file formats like deep shadow maps.

4 Geometry Types

Several volume primitive types exist, including voxel arrays, metaballs, and paged disk files. Volumes are implemented using a simple C++ API based on filtered evaluation to look up volume attributes. Rendering additionally uses an acceleration data structure based on a KD-Tree to cull empty space.

References