Rendering Tons of Sand

Chris Allen* Doug Bloom Jonathan M. Cohen Laurence Treweek
Sony Pictures Imageworks

1 Introduction

On *Spider-Man 3*, one of the villains is the Sandman, a character composed of sand. Sony Pictures Imageworks needed to render Sandman both as a solid form and as flowing sand grains, which in some scenes consisted of hundreds of millions of grains, and in some scenes the grains were close enough to the camera to see fine detail. This sketch describes our tools and workflow for rendering so many grains with RenderMan and controlling the level of detail for a consistent look.

(a) Close up shot showing fine grain detail.

(b) Distant shot with hundreds of millions of sand grains.

Figure 1:

2 Rendering

2.1 Grains on Polygons

In many of the shots Sandman was a solid object, animated as a deforming polygonal model. To save time and memory for simulating particles, our DSO could read files of polygons with settings indicating particle density and size and would generate the particles at render time, with a consistent method of birthing particles on polygons that guaranteed the particles would not pop on and off from frame to frame as the model deformed.

2.2 Buckets of Sand

To load the millions of sand grains with their shading parameters into memory would use more memory than our computers have, so we implemented a dicing scheme in our procedural DSO to take advantage of RenderMan's delayed evaluation of procedural bounding boxes. Our DSO would read the seed particles or the polygonal model, which would have attributes or global settings to indicate how many final renderable particles they would produce. The DSO then adaptively diced the scene's bounding box into smaller boxes, subdividing the boxes until they were below a threshold of particles that could be loaded simultaneously. Since RenderMan only loads procedurals as needed for a given bucket of pixels and it processes buckets from left to right, we computed the bounding boxes in screen space and diced the bounding boxes into horizontal strips until they were the size of a bucket. This allowed the renderer to discard particles it had finished with, leaving memory for the current bucket's particles.

2.3 Render-time Geometry Shaders

For dynamics-driven grains, we needed to simulate a manageable number of particles and then boost the number of particles at render time. We accomplished this through a combination of steps. We implemented a C-like shading language that lighters and shader writers could use to control particle attributes at render time, allowing them to birth new particles from the simulated seed particles and vary the new particles' size and other properties. Having an embedded scripting language also allowed the lighters and effects artists to create their own methods of birthing new particles, allowing them to birth disks of particles and even to birth curved disks of grains that conformed to the underlying polygonal mesh by inheriting the surface's curvature as a per-point attribute. We also wrote a preprocessing script to analyze the particles in a simulation and determine the minimum number of neighbors each particle has over the course of the shot, which was used to determine how many new particles would be birthed from each seed. This guaranteed that outlying particles would not be surrounded by clumps of rigidly attached particles.

2.4 Level of Detail

Since some shots were close-up shots of sand, we needed control for level of detail, so near grains would render with complex, modeled geometry while distant grains would render with simpler, less memory-intensive points. For greater control, we implemented our own level-of-detail scheme in the DSO rather than use the renderer's built-in system. Each grain was assigned a level of detail based on its size in screen space, with user-settable thresholds for the different grain types. The first level for the smallest grains consists of RiPoints. The second level for larger grains consists of "multi-points", clusters of four RiPoints for each rendered grain, with randomized normals to simulate four shading samples. The third level consists of RiCurves rendered as camera-aligned square patches, then displaced in a shader into the shape of grain geometry. The fourth and highest level simply stamped a rib archive of the polygonal sand grain model. This highest level was only used for extreme close-ups, and most grains were far enough from camera that a combination of points and multi-points was sufficient.

2.5 Tiling

Another technique to increase particle throughput was to tile the render into vertical strips so that the renders would still take advantage of the horizontal strips of screen-space dicing handled by our DSO, but also split the memory load of the rows of buckets. The tiles could easily be merged together in screen space, allowing us to render hundreds of millions of particles across multiple CPUs with minimal bookkeeping.

3 Results

Using the combination of techniques above, we were able to render a peak recorded count of 480 million particles in a single final frame in one of the shots. Figures a and b show example images from *Spider-Man 3*.

*e-mail: {callen,dougb,jcohen,laurence}@imageworks.com