8 Image Construction

8.1 Introduction

The process of lighting and shading a scene results ultimately in an internal representation of the surfaces in the scene with various data recorded about each surface. This data can include just about anything the user wants, but usually includes colour and opacity. Given a colour and opacity for each surface within the camera frame, we have enough information to produce a final image.

As the information in the scene is basically resolution free, producing a fixed resolution "snapshot" of this data is a sampling and filtering problem. This chapter discusses some appropriate methods of performing this sampling process, considers some inherent problems, and presents some widely accepted approaches to minimizing the effects of those problems.

To be able to effectively and efficiently sample the micropolygons passed through from the shading stage, we need to be able to gather some information from the micropolygons. This includes:

- *The screen-space bounding box of the micropolygon.* With this we can efficiently determine which micropolygons should be considered for which pixels in the final image. It should be kept in mind that any production quality scene is going to produce an inordinate number of micropolygons for us to sample, and this stage can easily become a major bottleneck in the rendering process; anything that can reduce the amount of work performed at the micropolygon sampling stage is very important. Testing if a micropolygon intersects a pixel can be very quickly accomplished using a bounding box. Once a micropolygon has passed this test, testing if the sample points lie within the bounding box will reduce the number of full micropolygon/point-sample intersection tests to a minimum.
- *The surface properties at a given sample point.* The micropolygon position information should be in screen space by now. Therefore the process of sampling the micropolygon involves first determining if the 2D sample point is within the 2D micropolygon – a simple point-in-quad test explained later. If so, then the position within the micropolygon is calculated, and the resulting u and v coordinates used to bilinearly interpolate the data stored at the corners. This amounts to Gouraud shading of the micropolygon, and as they are sufficiently small, any artefacts introduced by this approximation will not be visible.

- *The depth of the micropolygon at the sample position.* This is needed to allow us to store the sample information in the Z-buffer in the appropriate order. It needs to be quite accurate, so a simple bilinear interpolation will not suffice. The depth can easily be calculated using the normal to the micropolygon and the sample point, providing the depth information stored with the points is valid. Normally, when projecting 3D points into 2D, the depth information is lost. For our system, however, as the depth is important, the position information for the micropolygons are stored in a "hybrid" coordinate space. The x and y coordinates are in screen space, for the purpose of sampling. The z is in camera space for the use of the Z-buffer.

The micropolygons are sampled and the resulting data is stored in a depth-sorted list for each sample point. Once this stage is complete, the data needs to be compacted into a single resultant data value for each sample point. If the data is colour and opacity, this involves combining the colours, deepest first, using the opacity to filter the existing colour. This process is explained in detail later in the chapter. If the data does not include colour and opacity, then usually the closest sample to the camera is chosen.

Then the data needs to be further reduced to a single value per pixel. This is the process of filtering. The sample data are combined using a weighted filtering. It is often also necessary to consider samples from neighbouring pixels, thus the filtering stage is left until after the sampling and combining is complete for the whole bucket.

The sample data for each pixel that results from this entire process is the final data written to the image file, be it colour or some other user specified data.

8.2 Sampling

As explained in the section above, the capture of an image from our shaded scene is a sampling and filtering problem. The scene may contain varying frequencies of information. The contributing factors to these frequencies include the density of primitives in a particular section of the scene, the complexity of the surface detail of those primitives, and the complexity of the lighting and shading detail.

If the scene contains a surface with fine displacement mapping, that surface will produce a high frequency of surface data for the sampling stage to process. Similarly, complex surface shaders – those that use, for instance, highly detailed texture maps or fine procedural patterns – will also contribute high frequencies to the sample data. Anything that produces a rapid change in the colour visible to the camera in a small area of space will do the same.

To effectively sample the high frequency data represented by the scene and produce an image that closely represents the scene within the constraints of the image format, the sampler must be able to sample at a controlled frequency to reduce the errors due to aliasing. In most cases, simply sampling at each pixel will not be sufficient, since too much detail will be missed, and worse, if the scene is animated, the captured detail will change between frames resulting in a "twinkling" effect.