4 Shading

4.1 Introduction

Historically, the shading process was quite different when compared with modern renderers.

The original shader engines were primarily concerned with computing colours of points on a surface. A shader might take surface information like the position and normal to apply a lighting model which computed the resulting colour of the surface. A more sophisticated shader might apply an image-based texture to modulate the colour of the surface over space. At first glance, this seems very similar to what modern renderers do. However, these original shaders were typically hard-coded modules of the renderer which performed a specific task. They had a fixed number of parameters to control their behaviour. Parameters like diffuse colour or roughness could be passed to the shader to control how the resulting colour would be computed. In contrast, modern shading allows users to define their own parameters and their own algorithms for computing, not only colour, but also arbitrary data which gets output to the final image.

To give users more flexibility, some of the older engines broke the shading process up into several stages. One or more modules would be called to evaluate texture information which would be passed on to lighting modules to compute the colour for the surface. Still, the user was required to choose texture and lighting models from a fixed set of "materials".

The first step toward modern shading was presented by Rob Cook (1984) in his paper on "Shade Trees". Rather than provide large building-block modules for shading, Cook's shade trees were composed of very low level operations – things like adding two colours together, or normalizing a vector, as illustrated in Figure 4.1.

Though the shade trees were still comprised of modules which were connected by a graph, the shade trees provided a very flexible approach to shading since the user was able to compute the final surface colour in any fashion they wanted. Users weren't restricted to using a texture to modulate the diffuse component of a lighting model, they could encode arbitrary information into a texture and use it for any purpose in the shading computation. One of the unexpected uses cited by Cook was using textures to encode surface normals. These normals were used instead of the geometric normals to create surface detail. While this was only a small step from Blinn's bump mapping (Blinn, 1978), the shade tree model made

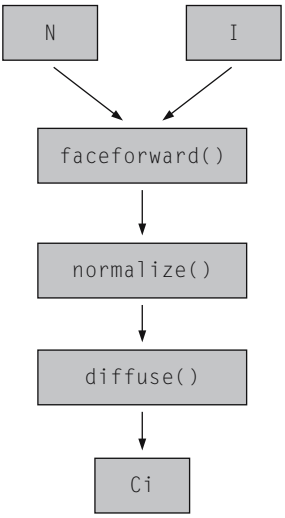

Figure 4.1 A simple shade tree.

it easy for users to experiment and do things that the author of the renderer could never have anticipated.

This was a turning point in shading. Previously, the shading engines of renderers were tightly coupled to the mathematical models used for lighting. However, Cook's shade trees provided users with a way of generating their own lighting models, generating their own textures, or basically, doing anything they wanted. Rather than choose from a fixed set of "materials" that the renderer supported, users could choose from a set of existing shade trees, or create a new shade tree from scratch to do what they wanted. The shading process, from the viewpoint of the renderer, was now to simply evaluate the shade tree and store the results.

By using shade trees, users could generate the first "false colour" images which have since become a standard production technique. Rather than generating images of illumination, users might choose to generate a false colour image of the surface normals, or any other required information. These false colour images could then be used for various purposes. Users might write custom programs that read the normal image as a source for other processing. They might use the orientation of the surface and position to generate hair or fur. More simply, they might use a false colour normal rendering to generate non-photorealitic images by using discontinuities in the normal map to determine silhouette edges. In older renderers, these techniques would have been virtually impossible as renderers did not have any "fixed" materials to generate the type of images the user needed.

One of the more popular renderers at the time was the Wavefront renderer. A feature of this renderer was that materials could be assigned on a per vertex basis and the renderer would blend between materials across a single primitive. Cook's approach was quite different in that the surface shader was applied on a per primitive basis. However, because of the flexibility of the shade trees, it was possible for a user to create a single shader which emulated the Wavefront renderer's feature