Check for updates

Shade Trees

Robert L. Cook

Computer Division
Lucasfilm Ltd.

Shading is an important part of computer imagery, but shaders have been based on fixed models to which all surfaces must conform. As computer imagery becomes more sophisticated, surfaces have more complex shading characteristics and thus require a less rigid shading model. This paper presents a flexible tree-structured shading model that can represent a wide range of shading characteristics. The model provides an easy means for specifying complex shading characteristics. It is also efficient because it can tailor the shading calculations to each type of surface.

CR CATEGORIES AND SUBJECT DESCRIPTORS: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; E.1 [**Data Structures**]: *Graphs, Trees.*

ADDITIONAL KEY WORDS AND PHRASES: color, computer graphics, illumination, lighting, reflection, shading, shadows, texture

1. Introduction

Making synthetic images look realistic is an important goal in computer imagery for two reasons. First, some applications require a high degree of realism as an end in itself. Second and more generally, realism acts as a measure of our techniques and understanding. To the degree that we lack the ability to make pictures look realistic, we also lack some artistic control.

Making a realistic image involves solving a number of different problems. This paper addresses the problem of shading, or selecting colors for points on each surface, and more specifically the problem of controlling and directing the shading calculations. Other problems, such as constructing a model and animating it, are equally important to realistic image synthesis but are not addressed in this paper.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1984 ACM 0-89791-138-5/84/007/0223 $00.75

At the heart of the shading calculations is the simulation of the way light interacts with objects. Early work in reflection models was done by Henri Gouraud[10] and by Phong[16], with more accurate models being developed by Jim Blinn[2] and by the author[7], who applied the shading model to the simulation of specific materials. Blinn has developed a separate shading model for clouds[5]. Turner Whitted included reflection and refraction[18].

Textures allow us to map shading properties onto a surface mathematically, greatly increasing the visual complexity and richness of an image without the overhead of explicitly modeling those properties. Texturing was first used in computer graphics by Ed Catmull[6]. Jim Blinn later extended the use of texturing to surface bumps, roughness, and reflections[1, 4, 3]. Geoff Gardner included texturing of transparency[9].

The trend in shaders has been toward more flexibility and generality, as evidenced by Blinn's generalization of texturing[1] and Whitted's shader dispatcher[19]. What has been lacking is an overall system that integrates the various shading and texturing techniques. This paper introduces such a system, one that is based on a more general approach to shading. The new approach provides a language for describing surfaces and allows traditional shading techniques to be combined in novel ways.

Previous shaders have been limited by the use of fixed models of light reflection into which all surfaces must be fit. The new approach is modular and assumes that no single shading model is appropriate for all surfaces. In some cases utter simplicity is desired, while in others we may require a complexity that would normally be a burden. Because of its modular nature, the new shader can handle both of these extremes in the same image; it performs only the calculations needed for the simple cases while allowing arbitrarily complex calculations where they are required.

2. Appearance Parameters

A number of different geometric, material, and environmental properties together determine the color of a surface. Any value that is used in the shading calculation is called an *appearance parameter.* Appearance parameters include the surface normal, the color of the light source, the shininess of the surface, bump maps, etc.

The traditional approach to shading is to divide the calculations into two stages:

1. Determining the values of the appearance parameters.
2. Using those values to evaluate the fixed shading equation.

This approach can offer a great deal of generality in the first stage but is inflexible in the second. Appearance parameters may be determined in a number of ways, including texture mapping and normal interpolation. The shading equation itself, however, is fixed. All surfaces must be fit into it, no matter how complex and no matter how simple. Little allowance is made for the extremely diverse ways in which objects interact with light.

3. Shade Trees

A more general approach is to eliminate the fixed shading equation and the entire two stage approach. Rather than attempt to describe all possible surfaces with a single equation, the shader orchestrates a set of basic operations, such as dot products and vector normalization. The shader organizes these operations in a tree.

Each operation is a node of the tree. Each node produces one or more appearance parameters as output, and can use zero or more appearance parameters as input. For example, the inputs to a "diffuse" node are a surface normal and a light vector, and the output is an intensity value. The normal might come from the geometric normal, a bump map, or a procedural texture. The output might be the input to a "multiply" node, which would multiply the intensity by its other input, a color.

The shader performs the calculations at the nodes by traversing the tree in postorder. The output of the root of the tree is the final color. Basic geometric information, such as the surface normal and the location of the object, are leaves of the tree. (In general, the nodes actually form a directed acyclic graph, because a single appearance parameter can be used as input to more than one node.)

Even an appearance parameter that is usually thought of as the final shade can itself be treated as an intermediate step. This is particularly useful in rendering a surface that consists of different materials. The final shade can be a combination of the shades of the various materials, with the amount of the various materials based perhaps on a texture map.

Shade trees can describe a wide range of shading situations from simplest to the most complex. Different types of shading calculations can coexist in a single image, with each surface using as many or as few operations as it requires.

4. Light Trees

The appearance parameters used in the shading calculations include the light source direction and color. These appearance parameters are described by their own tree. Light trees are separate from shade trees so that each light tree can be grafted onto several different shade trees.

Different types of lights require different calculations[17, 11]. The intensity of a local light source changes as the square of the distance from the light. Spotlights have a goniometric curve that describes their intensity as a function of direction. Some lights have flaps that abruptly restrict their illumination. All of these lights are easily described by light trees. For example, the inputs to the "spotlight" tree are the direction of the central axis of the light beam and the rate at which the intensity of the beam decreases with angle, in addition to the position of the light and the location of the point being illuminated. It uses the relevant formulas to calculate the intensity of the light at that location. These calculations, which are so specific to this one particular type of light source, are isolated from the rest of the shading, communicating only through the appearance parameters.

5. Atmosphere Trees

The final output of a shade tree is the *exitance*, the color and intensity of the light leaving the surface. But this is not necessarily the same color and intensity as the light that reaches the eye. Atmospheric effects are described by a tree that has the exitance as one of its inputs and the light actually reaching the eye as its output.

Atmospheric effects are often described by procedural models. For example, haze is an exponential function of distance[14, 13] and can vary with direction. Loren Carpenter simulated sky and haze in his film *Vol Libre* and developed a general atmosphere model for the Genesis sequence in *Star Trek II.* These models are easily incorporated into atmosphere trees.

Rainbows can also be described by an atmosphere tree, with light being added in the primary and secondary bows and subtracted in Alexander's dark band[15, 12]. The color and intensity are a function of the angle between the light direction and the viewing direction.

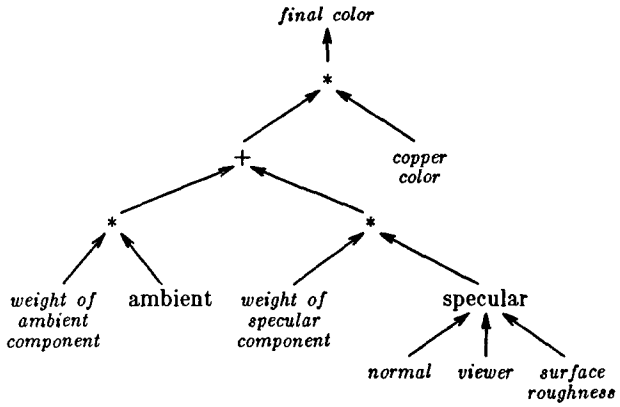


Figure 1a. Shade tree for copper.

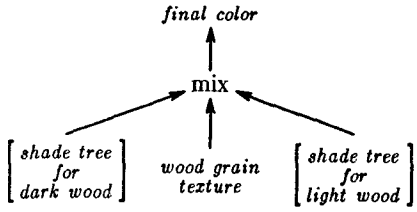


Figure 1b. The mix node in a shade tree for wood.

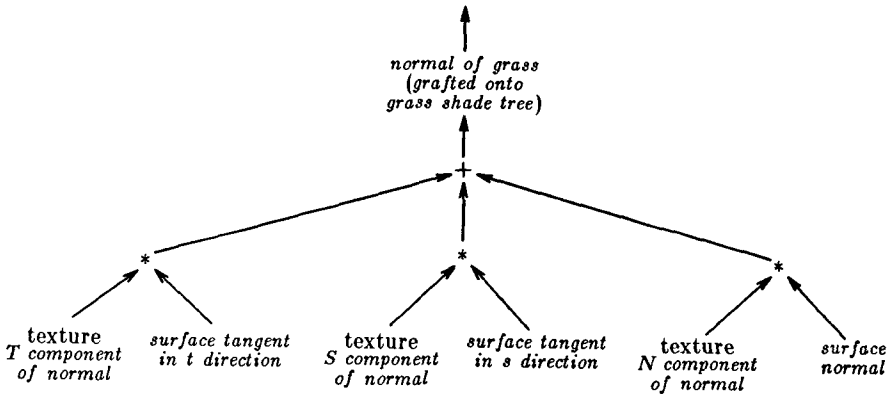


Figure 1c. Textured grass normal.

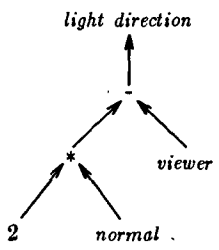


Figure 1d. "Highlight at" branch of a light tree.

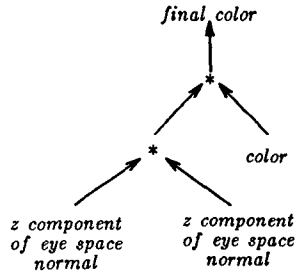


Figure 1e. Simple shade tree.

6. Implementation

To facilitate describing shade trees and building up a library of surfaces, we have developed a special shade tree language. A program written in this language is compiled into an internal representation of a shade tree. Programs in this language are also used to describe light trees and atmosphere trees.

A number of keywords, such as *normal* and *location*, refer to the basic geometric information provided as input to the shader. The final results are referred to by the keywords *final_color* and *final_opacity*.

Several types of nodes are built into the language, including mathematical functions such as *square root* and *normalize* and shading functions such as *diffuse* and *specular*. Other more specialized nodes can be added dynamically; when such a node is declared in the language, the shader searches for a file by that name and loads it. This provides enough flexibility to add new and exotic shading functions easily.

Variables in the language represent appearance parameters, and statements describe how to connect the nodes with appearance parameters. For example, the following program describes metallic shading and defines the shade tree shown in Figure 1a. Note that the ambient and specular nodes are built into the language, and that the output of the light trees is available to all nodes.

```
float a=.5, s=.5 ;
float roughness=.1 ;
float intensity ;
color metal_color=(1,1,1) ;
intensity = a*ambient() +
        s*specular(normal,viewer,roughness);
final_color = intensity * metal_color ;
```

Point variables can be specified in eye space or in world space, whichever is more convenient. World space coordinates are indicated by preceding them with the keyword "world_space".

This language works in conjunction with a modeling language to associate surfaces and light sources with objects. A light can be assigned to the group of objects because some lights affect only part of the environment. Its light tree calculations are performed only for the objects it affects.

The *surface* command in the modeling language designates a shade tree for an object. The values of variables in the surface language can be overridden here. For example, for the above "metal" surface, the statement

```
surface "metal",
        "metal_color", material bronze,
        "roughness", .15
```

in the modeling language initializes the variable "roughness" to be .15 instead of the default .1 and "metal_color" to be the color of bronze instead of white.

7. Experience with Shade Trees

This section presents several specific examples of shade trees and discusses the benefits of this new way of thinking about shading. Since we first started using shade trees, we have discovered many more uses for them than originally expected.

One surprise was the new uses of textures. For example, we rendered some leaves of grass generated by Bill Reeves by creating a texture map of transparency that could be mapped onto a polygon. Instead of using the texture to store the color of the blades for a particular orientation relative to the light source, the surface normal can be encoded in the texture and used in a shade tree as shown in figure 1c. The shading uses the correct normal and changes appropriately as the lights move.

Shadows, including penumbras, can be calculated or painted ahead of time and stored as textures that are accessed by the light tree. The ambient light is a separate light source; it is usually a constant, but it can also be textured to account for the dimming of the ambient light in corners.

Perhaps the most useful shade tree node has been the "mix" node, which uses one of its inputs to interpolate between the other two. This can be used to select between two types of materials, so that a pattern of one material can be inlaid into another. The mix node can also be used for a single material that is not homogeneous, such as wood. Many types of wood have a grain pattern of a light and dark wood. The light and dark wood are really separate materials, with separate sets of appearance parameters such as color and shininess. The grain is a single channel of texture that selects between these two materials. We compute the color of light oak and the color of the dark oak and then mix the two based on the grain texture. Figure 1b shows how the mix node is used in a shade tree for wood.

Metal fleck paint has flecks are oriented in random directions about the surface normal. A special node generates the location of each fleck on the surface and the orientation of each fleck relative to the surface normal. We add this relative normal to the surface normal and renormalize to get the true fleck normal, which is used to shade the fleck. The final color of the surface is a mixture of the color of the base paint and the color of the flecks, based on the procedural texture for the location of the flecks. Because the reflection from the flecks is highly directional, the "mix" node is essential. We can not simply shade a blend the appearance parameters (including the normals) of the flecks and the paint.

The input to the "texture" node is a set of texture coordinates. Texture coordinates are traditionally the same as the object's natural coordinates u and v. But once we regard the texture coordinates as an appearance parameter, we see that they do not need to be identical to u and v. We call the texture coordinates s and t to distinguish them from patch coordinates u and v. If we choose s and t properly, a single texture can extend over several patches without seams.

One of the more exotic uses of shade trees is an extension to bump maps called *displacement maps*. Since the location is an appearance parameter, we can actually move the location as well as perturbing the surface normal. Displacement maps began as a solution to the silhouette problems that arise when bump maps are used to make beveled edges. They are useful in many situations and can almost be considered a type of modeling. This use of shade trees, however, depends on performing the shading calculations (or at least the displacement map part of them) before the visible surface calculations.

In many cases, we are interested not in the actual location of a light source, but in the position of its highlight on a particular surface. The position of the desired highlight can be an input to the light tree, which calculates the light direction that would make a highlight appear at that given location. The tree for this calculation is shown in Figure 1d. It has proved useful in setting up the lighting for a scene.

Unusual shading functions can added to the library of shades easily. "Cat's eye" reflectors on highways reflect light back toward the light source. They are essentially a specular reflection with the normal pointed toward the light source. Marble has a textured diffuse component and a mirror-like specular component. The glowing shock wave in the Genesis sequence in *Star Trek II* was rendered by Loren Carpenter using a special purpose shading function he developed. This function was later easily described as a shade tree.

Other shade trees are used just for debugging. For example, a shade tree that assigns each patch a different random color can be useful in detecting bugs in the patch splitting code. The surface normal can be encoded in the color to look for discontinuities. It is easy to use a simple shading model, such as the one shown in Figure 1e, for trial images and to switch to more elaborate calculations for the final image.

Intermediate results can be computed by one shade tree and stored in a texture for later use by another shade tree. This is useful in calculating shading information that does not change from frame to frame within the scene.

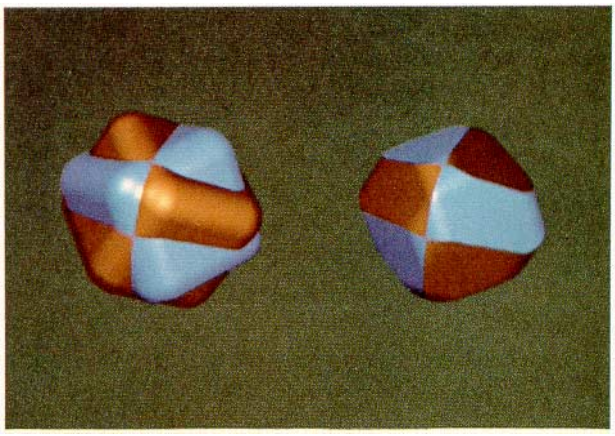


Figure 2. Union and Intersection of Two Cubes Beveled With Displacement Maps.

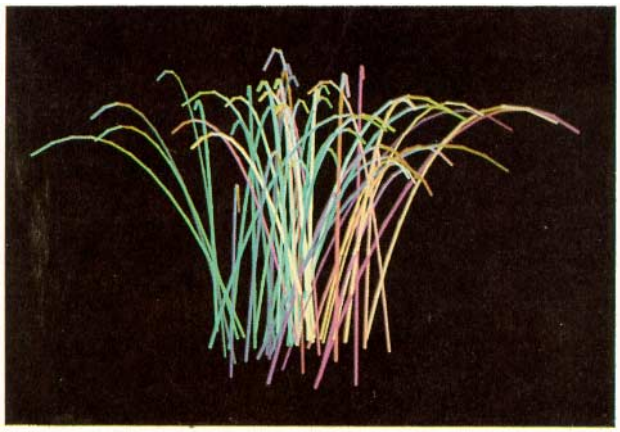


Figure 3a. Grass Normal Texture Map.

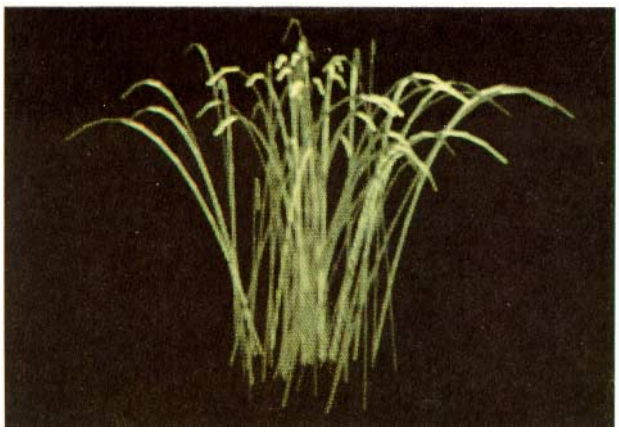


Figure 3b. Grass Rendered with Textured Normals.

8. Examples

Figure 2 shows the union and intersection of a plastic and a copper beveled cube. Each cube is described by 6 polygons that are beveled in the shader according to a procedural texture. The beveling is a displacement map that moves the locations as well as the normals.

Figure 3a is a grass texture map generated Bill Reeves. This texture was used to generate Figure 3b, which consists of a single polygon with texture mapped transparency. In addition, the red, green, and blue components of the texture are used to encode the three components of the normal relative to the normal of the surface. The resulting picture has highlights that are appropriate to the local lighting environment.

Figure 4 is *Road to Point Reyes* [8] The road lines, asphalt, and oil spots are each described by shade trees; the outputs of these trees are grafted to a "mix" node and mixed according to a texture map. The wooden fence posts and bronze chain links are described with shade trees. The hills are rendered with a three channel color texture map generated procedurally by Tom Porter. The rainbow is described by an atmosphere tree. During the early stages of design, texture maps were used to render the grass, the bushes, and the puddles quickly; these were later replaced by more exact models.

Figure 5c, *Bee Box,* illustrates light trees, displacement maps, and the "mix" node. The light is a spotlight, and its shadow (including penumbra) is produced by a texture map in a light tree. The regions of wood, ivory, copper, and bronze are selected by three channels of the texture map shown in figure 5a. Another channel controls the surface roughness in the copper and bronze regions. By contrast, figure 5b shows the same box rendered with diffuse blue shading and no displacement maps or shadows.

9. Efficiency

The overhead involved in using shade trees is small since the tree construction and traversal is done ahead of time by the shade tree compiler. At run time there is just a list of routines to call for each surface and a list of arguments (i.e., appearance parameters) for each routine.

Some of the surfaces described by shade trees are complex, and the shading time increases with the complexity of the shade tree. Shade trees are very useful in optimizing the shading calculations, however, because it is easy to adjust surface descriptions to the appropriate level of computation. If a surface is perfectly diffuse, the specular shading calculations are never used. If the geometrical attenuation of the Torrance-Sparrow[2] shading model is not necessary for a particular surface, it can easily be avoided. Reflections can be calculated with a "trace a ray" node or with an an environment map, as appropriate. Color maps, bump maps, or displacement maps can be used depending on the distance to the object.

Notice that in *Bee Box*, the wood uses only one channel of texture (the amount of grain) instead of the three one would expect (red, green, and blue). This one channel controls an entire set of appearance parameters, including color and roughness. Since wood is a mixture of surfaces, based on a texture map, only one branch of the shade tree need be descended in places where the texture calls for only one of surfaces.

10. Conclusions

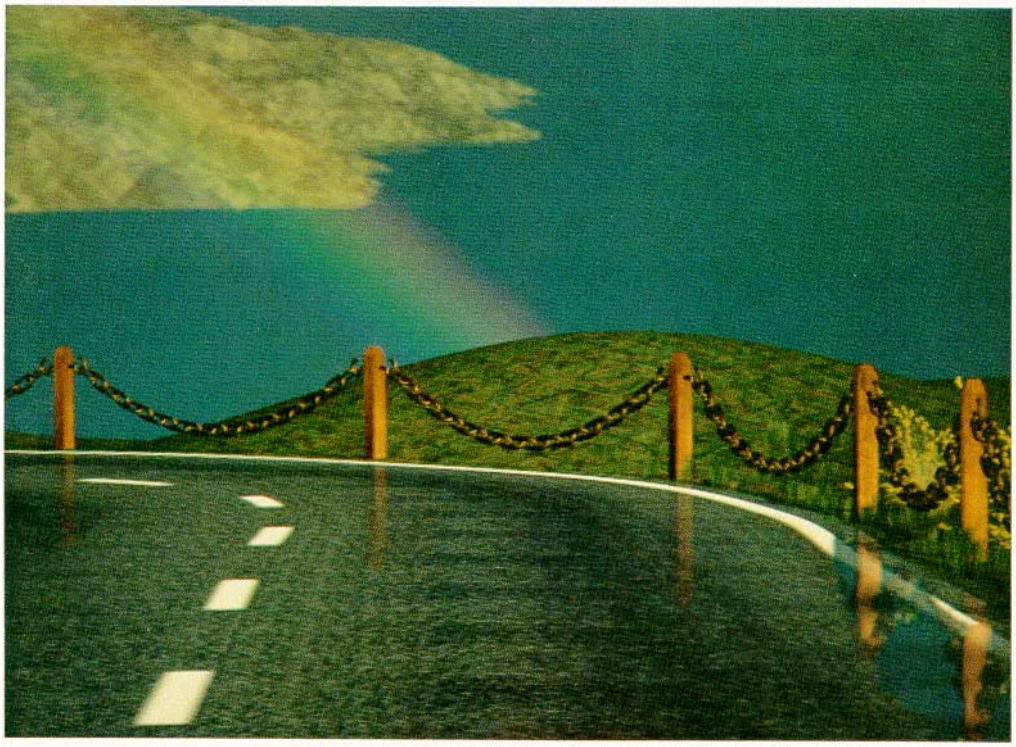
Shade trees offer a way to specify and change shading properties quickly and easily. They are flexible because they are not based on a fixed shading formula; instead they provide a general way to connect basic shading operations. They are efficient because they customize the shading calculations for each type of surface.

11. Acknowledgements

Many of the ideas in this paper came out of discussions with Loren Carpenter. In some cases it is hard to say exactly who thought of what, because many of the ideas came out in the course of brainstorming sessions. Our discussions included displacement maps and shadow textures, which led to the extension of shade trees to light trees.

Tom Duff provided the nugget of code (a run time loader) that inspired a flexible implementation of shade trees. The modeling language that provides all of the hooks for lights and surfaces was written by Bill Reeves and Tom Duff. John Lasseter painted the texture of the bee. Discussions with Dan Silva were helpful in the early stages. This work began as a continuation of work done at the Program of Computer Graphics at Cornell University.

Figure 4. Road to Point Reyes.



References

1. BLINN, JAMES F. AND MARTIN E. NEWELL, "Texture and Reflection in Computer Generated Images," *Communications of the ACM*, vol. 19, pp. 542-547, 1976.
2. BLINN, JAMES F., "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics*, vol. 11, no. 2, pp. 192-198, 1977.
3. BLINN, JAMES F., "Simulation of Wrinkled Surfaces," *Computer Graphics*, vol. 12, no. 3, pp. 286-292, August 1978.
4. BLINN, JAMES F., "Computer Display of Curved Surfaces," PhD dissertation, University of Utah, Salt Lake City, 1978.
5. BLINN, JAMES F., "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces," *Computer Graphics*, vol. 16, no. 3, pp. 21-29, July 1982.
6. CATMULL, EDWIN, "A Subdivision Algorithm for Computer Display of Curved Surfaces," Phd dissertation, University of Utah, Salt Lake City, 1974.
7. COOK, ROBERT L. AND KENNETH E. TORRANCE, "A Reflection Model for Computer Graphics," *ACM Transactions on Graphics*, vol. 1, no. 1, pp. 7-24, 1982.
8. COOK, ROBERT L., LOREN CARPENTER, THOMAS PORTER, WILLIAM REEVES, DAVID SALESIN, AND ALVY RAY SMITH, "Road to Point Reyes," *Computer Graphics*, vol. 17, no. 3, July 1983. title page picture
9. GARDNER, GEOFFREY Y., EDWIN P. BERLIN JR., AND BOB GELMAN, "A Real-Time Computer Image Generation System Using Textured Curved Surfaces," *The 1981 Image Generation/Display Conference II*, pp. 60-76, June 1981.
10. GOURAUD, HENRI, "Computer Display of Curved Surfaces," PhD dissertation, University of Utah, Salt Lake City, 1971.
11. HALL, ROY A. AND DONALD P. GREENBERG, "A Testbed for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, vol. 3, no. 8, pp. 10-20, November 1983.
12. HULST, H. C. VAN DE, *Light Scattering by Small Particles*, pp. 228-266, Dover, New York, 1957.
13. MCCARTNEY, EARL J., *Optics of the Atmosphere*, pp. 1-49, John Wiley & Sons, New York, 1976.
14. MINNAERT, M., *The Nature of Light and Color in the Open Air*, Dover, New York, 1954.
15. NUSSENZVEIG, H. MOYSES, "The Theory of the Rainbow," *Scientific American*, vol. 236, no. 4, pp. 116-127, April 1977.
16. PHONG, BUI TUONG, "Illumination for Computer Generated Pictures," *Communications of the ACM*, vol. 18, pp. 311-317, 1975.
17. WARN, DAVID R., "Lighting Controls for Synthetic Images," *Computer Graphics*, vol. 17, no. 3, pp. 13-21, July 1983.
18. WHITTED, TURNER, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, vol. 23, pp. 343-349, 1980.
19. WHITTED, TURNER AND DAVID M. WEIMER, "A Software Testbed for the Development of 3D Raster Graphics Systems," *ACM Transactions on Graphics*, vol. 1, no. 1, pp. 44-58, January 1982.

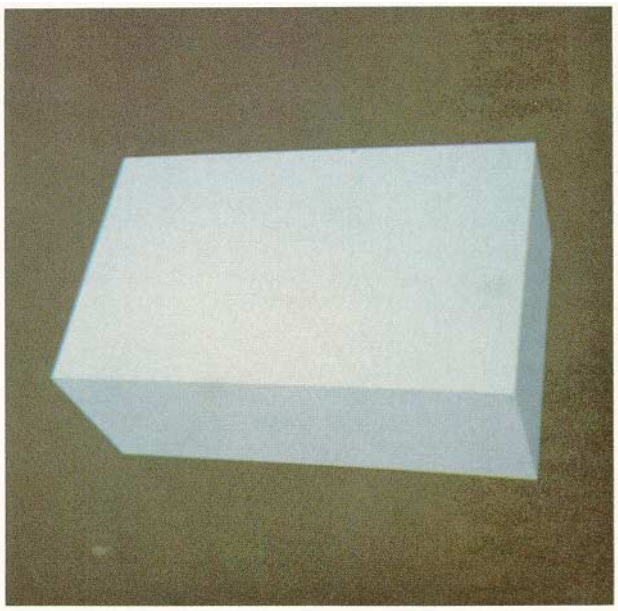


Figure 6a. Plain Box.

Figure 6b. Bee Texture.

Figure 6c. Bee Box