Classical ray-tracing techniques, which have produced the most realistic computer-generated images to date, are being enhanced in this developmental system.

A Testbed for Realistic Image Synthesis

Roy A. Hall and Donald P. Greenberg
Cornell University

Our goal in realistic image synthesis is to generate an image that evokes from the visual perception system a response indistinguishable from that evoked by the actual environment. This objective requires the ability to model the physical behavior of light, specifically within the visible range, as it is propagated through an environment. Once the physical behavior is modeled, we must then display the result in such a manner that it is perceived correctly.

We can describe the propagation of light through an environment, using mathematical models of the physical laws governing electromagnetic radiation. This description of light propagation is independent of viewer position. In generating an image of the environment, we wish to "capture" the electromagnetic events, specifically light energy, at an arbitrary image plane and display this for a viewer. For example, a camera is commonly used to record the events at an image plane on film, which can then be projected for viewing. In computer graphics, we can bypass the camera, and even the physical environment, and instead simulate the way light would reach an image plane from a modeled environment.

In the past, image generation schemes computed images so that they could be sent directly to a display device. The processes of light propagation, image capture, and image display, as well as the perceptual considerations of the viewing environment, have traditionally been lumped into a single computational operation. However, image synthesis should be addressed as a multistep process because the problems and constraints associated with display resolution and color reproduction are independent of the methods used for the modeling of light propagation and the determination of the intensity at the image plane.

The initial step in the process is the modeling of the physical environment, including object geometry, object positions and orientations, material characteristics and surface finishes, and lighting attributes. Although the model geometrically describes a physical environment, it is not necessary for the environment to actually exist.

The second step is the simulation of light propagation through the environment. The light energy incident on any surface of the environment is described as a summation of light coming directly from light-emitting sources and light reflected from other surfaces in the environment. The indirect light energy is often referred to as "global illumination information." Once the global illumination information for the environment has been determined, it should be possible to determine the intensity of the light leaving any surface in the environment in any specified direction.

The third step is to determine the intensity function at the image plane (Figure 1). If an image plane could be instrumented so that the spectral intensity moving through the plane toward a viewer could be sampled at any point, it would be possible to map the energy that reaches the eye through the image plane as a function of the location on the plane (x,y) and the wavelength (λ). We refer to this mapping as the intensity function at the image plane. In a real scene, this function is continuous in both the spatial and intensity domains. In computer graphics, the computation of intensity at the image plane should

0272-1716/83/1100-0010$01.00 © 1983 IEEE

reduce to projecting small discrete areas on the image plane into the environment and integrating the intensity of the visible objects within the projection of each differential area to determine the corresponding intensity. Prior to this computation, it is obviously necessary to define the viewing conditions, including viewer position, view direction, and field of vision.

The fourth step is the conversion of the discrete image plane intensity information into a displayable form, culminating in the display of the image. This step can be subdivided into a number of operations, including transformation to the display pixel resolution, filtering to eliminate aliasing, adaptation to the color limitations of the display device, and conversion to the appropriate red, green, and blue components. The distinction between the intensity function generation process and the display process is important; thus, we discuss display operations as a separate step in the total image generation process.

This multistep process has been implemented in the system we describe in this article—a testbed system for the generation and display of realistic images, developed in Cornell University's computer graphics program. Currently, the system uses ray tracing for image generation, with a number of enhancements to previous ray-tracing schemes, but the flexibility of the system structure would allow the use of other rendering methods.

Historical overview

The methods applied to realistic image synthesis, as well as the connotation of the term "realistic" in computer synthesized images, have changed greatly in the past two decades. Many of these changes can be attributed to advances in hardware as well as in software. Increased computational speed of equipment and technologies that allow both high spatial resolution and high color resolution have made possible the generation and display of very sophisticated images.

Early approaches to realistic image synthesis with raster displays primarily involved the visible surface determination of polygons, not the determination of their color. Examples of such algorithms are those by Schumacker et al.,[1] Newell et al.,[2] Warnock,[3] Watkins,[4] Weiler and Atherton,[5] and Sechrest and Greenberg.[6] A comprehensive review of this type of algorithm appears in "A Characterization of Ten Hidden-Surface Algorithms" by Sutherland et al.[7] In all of these efforts, emphasis was placed on the determination of the portion of the environment that should be displayed, rather than on the light reflection model and the realism that is generated by color and texture.

Many of these algorithms used the reflection from a Lambertian diffuse surface to determine the color of the displayed polygons. Warnock used a reflection model that included both an object color term and a specular term. Gouraud[8] proposed a method of color interpolation across polygonal approximations that would generate the appearance of smoothly curved surfaces. Phong[9] proposed an alternate method that interpolated surface normals across each polyon and applied a reflection model for the determination of the color of each pixel as a function of the direction of the surface normal.

The Phong reflection model, an empirical approximation of observed data, has been particularly significant in the evolution of realistic image synthesis methods. Its formulation includes a diffuse term that provides surface color and shading, and a specular term that provides realistic highlights from direct light source reflections. The model is very easy to implement and is still widely in use. Additionally, Phong recognized that to achieve realistic appearance, surface normals must be mapped to every pixel before the application of the reflection model.

Basing them upon the Torrance-Sparrow reflection model,[10] Blinn[11,12] suggested less empirical improvements of the Phong reflection model. His consideration of the specular component of the reflection led to the observation that, for mirror-like surfaces, the magnitude of the specular component is related to the intensity that reaches the surface from the mirror direction. He also proposed a number of methods for providing realistic textured appearance, which greatly increased apparent scene complexity and, subsequently, increased scene realism.

Cook and Torrance[13] proposed a reflection model that describes the behavior of light in terms of energy equilibrium and electromagnetic wave theory, based upon the Beckmann scattering model.[14] Application of this model results in a very realistic image appearance of a wide variety of materials with varied surface finishes. Unfortunately, the model requires spatial integration of the global illumination information to provide the incident energy on a surface. None of the present methods of image synthesis can generate the information required for application of this model to situations other than an isolated object suspended in space.

Perhaps the first use of global information in calculating intensities for image generation was accomplished by

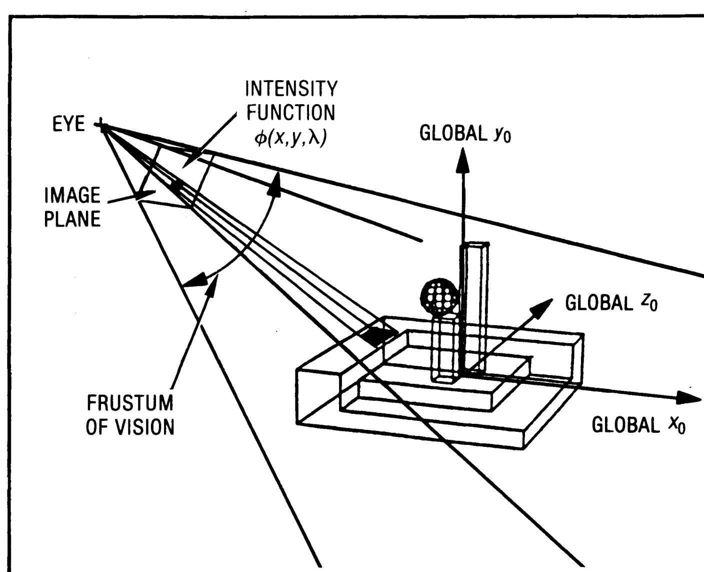


Figure 1. Intensity function at the image plane.

mapping an environment onto a global hemisphere.[15] Intensities were computed from a look-up table that provided intensity information from any direction in the environment. This method was adequate for describing the reflection of an environment distant from the object being rendered, but could not provide information for the interaction of two objects in close proximity.

A more appropriate and general approach is ray tracing. Ray tracing was suggested by Appel[16] and later used by MAGI[17] to solve the hidden surface problem. In attempts to solve the global illumination problem, the methodology was extended by Kay[18] and by Whitted[19] in his classic paper. Ray tracing is used to determine the global illumination information relevant to the image plane.

The results of ray tracing have been some of the most realistic images to date. However, there are many scenes that cannot be adequately modeled by ray tracing, and there are several shortcomings to the methods that have been implemented. First, the reflection models are empirical and have not accounted for the theoretical behavior of light and the required energy equilibrium conditions. Second, the ray-tracing method provides only point-sampled information, which, in addition to causing aliasing problems, is not sufficient for the application of energy equilibrium models to light behavior. Third, the method combines the modeling of global illumination and the computation of the image plane intensity function into a single operation. Fourth, the method of application of the reflection model has resulted in a loss of spectral information at the image plane. Thus, despite the impressive images, there are still many improvements to be made.

Implementation of a testbed system

Components of an image synthesis system must perform the following tasks:

(1) environment modeling,
(2) assignment of viewing parameters (setup operation),
(3) simulation of light propagation through the environment,
(4) calculation of the image plane intensity function (image generation), and
(5) display of the image plane intensity function (image display).

A testbed imaging system (diagrammed in broad outline in Figure 2) was implemented in the Program of Computer Graphics at Cornell University to perform these tasks. The main goal in the development of this graphics testbed was a flexible, modular system that would impose a minimum of constraints on future research. The system separates the imaging process into a series of functional processes that communicate through the use of files. The file structure has been designed to assure flexibility and the retention of all required image

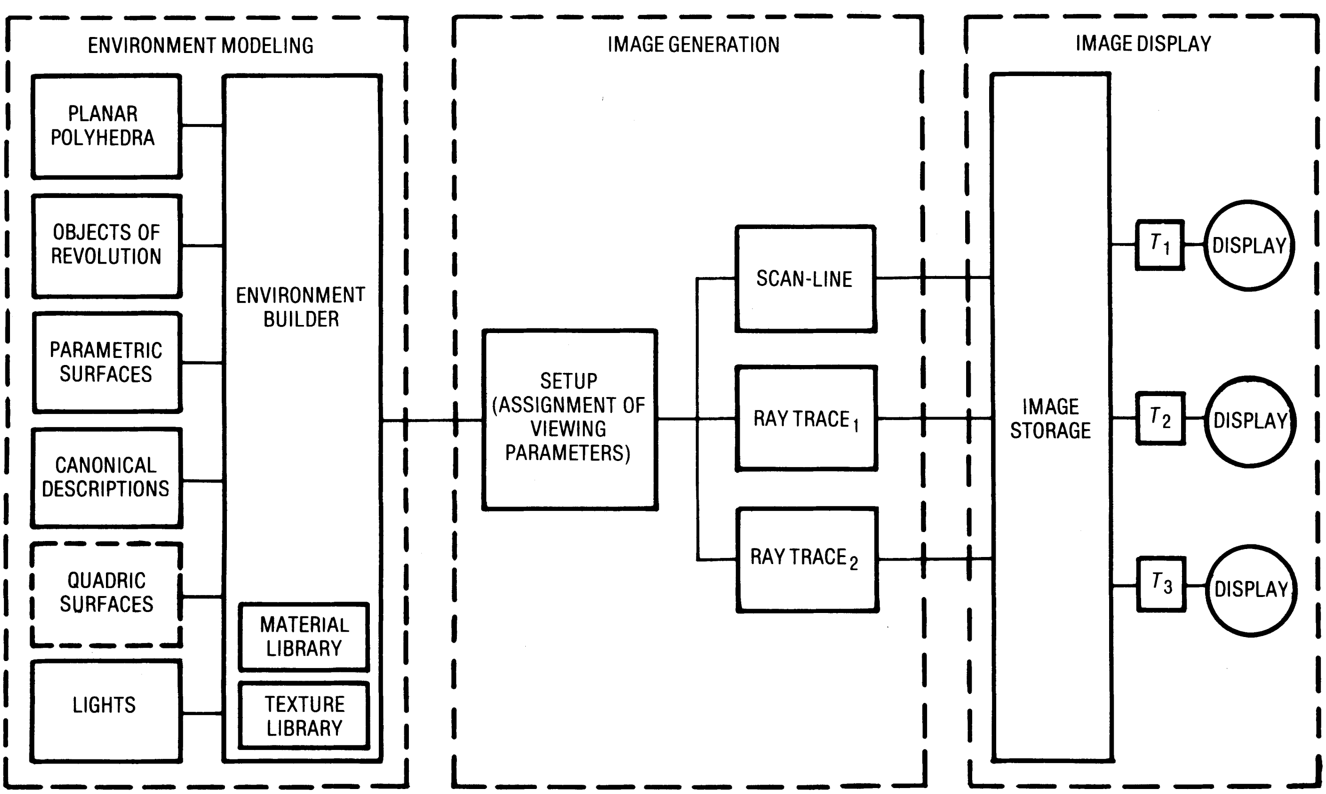


Figure 2. Testbed image synthesis system of the Cornell Program of Computer Graphics.

information. In the system's first generation, only ray-tracing routines have been implemented, although it could use any type of visible surface or image generation algorithm.

We have attempted to use ray tracing to perform all the required light propagation modeling during the calculation of the intensity function, thus combining the third and fourth operations of the steps listed above into a single operation. Although this methodology fails to address some of the issues surrounding the modeling of global illumination, it does have the capability to produce very realistic images and can serve as an integral part of a system that would allow exploration of alternative methodologies.

Environment modeling. Individual objects are separately created by means of a number of geometric input routines. Separate routines exist for the creation of complex planar polyhedra, objects of revolution, parametric surfaces, canonical descriptions, and quadric surfaces. Furthermore, any geometric modeling system can be used as an input routine to the imaging system, provided that the bounding surfaces are accurately defined and that attributes can be associated with each discrete surface.

A unique type of input routine is used for the modeling of light sources.[20] In addition to standard geometric information, the program allows the user to define spectral energy distribution and directional intensity of the light source.

The environment builder consists of a set of editing routines that perform two basic functions. The first is the positioning of created objects into a global environment description. All the scaling, translation, and rotation options are available to accurately place objects in relation to one another. The second function is the assignment of attributes to the object surfaces. These include surface finish characteristics, such as roughness and root-mean-square slopes, and wavelength-dependent properties, such as material spectral reflection curves and texture patterns.

Viewing and image parameters. A setup program provides the additional information required for ray tracing, such as viewer position, image resolution, maximum depth of the intersection tree, cutoff threshold for the intersection tree, pixel subdivision criteria, spectral sampling criteria, and the reflection model type.

Ray tracing. A ray is traced from the eye through each pixel into the environment. At each surface which is struck by the ray, a reflected and/or refracted ray can be spawned (Figure 3). Each of these must be recursively traced to establish what surfaces they intersect. As the ray is traced through the environment, an intersection tree is constructed for each pixel (Figure 4). The final pixel intensity is determined by traversing the tree and computing the intensity contribution of each branch according to the reflection model.[19]

The intensive computational operations in a ray-tracing system are the intersection tree building and the traversal of the tree for the intensity calculation. Testing for shadows has been found to be very expensive computationally, particularly for environments that contain complex lighting situations.

The intersection tree consists of branches that represent the propagation of a ray through the environment, and of nodes that represent the intersection of rays with interfaces in the environment (Figure 4). In the implementation used for this investigation, the intersection tree is a binary tree implemented as a forward- and backward-linked list. This implementation was chosen because the tree is generally neither full nor complete, and this scheme simplified the packing of used nodes into a contiguous memory space. The backward link is not required in the building of the tree or in the traversal of the tree for color computation, but it is often helpful in the debugging process.

In this procedure, a dummy node is created as the parent node to the root node of the intersection tree. This dummy node can be thought of as representing the eye. The intersection location and direction cosines of the dummy node define the view vector for the sample point.

The data block representing a node on the intersection tree must contain sufficient information to fully describe the environment at that node. This includes information such as the material at either side of the interface, the surface normal, the reflected vector, the refracted vector, and the distance from the previous node.

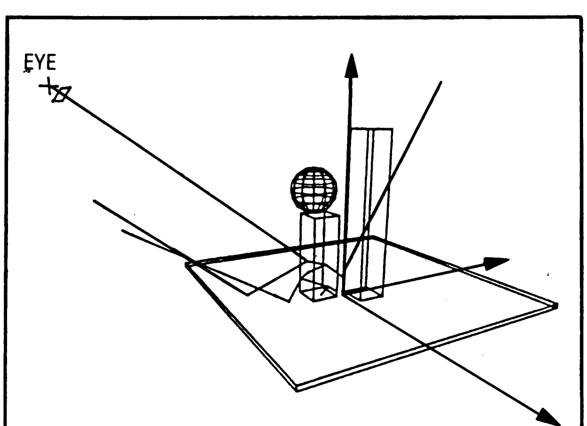


Figure 3. A ray traced through an environment.

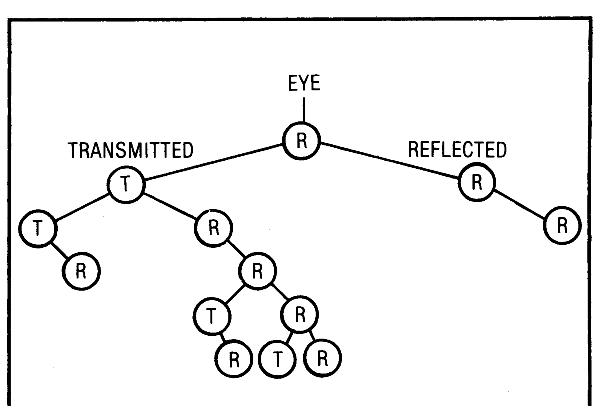


Figure 4. Intersection tree for a ray traced through the environment.

The intersection procedure controls the search through the objects in the environment to determine which object is first encountered by a ray. This routine calculates intersections for all object and light source entities.

The reflection model is applied during the traversal of the intersection tree to generate an intensity for the current sample point. In this implementation of ray tracing, it was desirable to support a number of alternate reflection models to allow easy modification and comparison. We accomplished this by establishing common calling conventions for all reflection models and loading the appropriate routine addresses into a branch table that was used in the tree traversal procedure.

The intersection tree is traversed from the bottom up in the process of generating the intensity at the sample point on the image plane. The intensity information for the children nodes of the node currently being evaluated is retained, and the reflection model is applied to generate the intensity at the current node. Once the recursive tree traversal procedure has been executed, the intensity for the sample point is contained in the root node data block.

In actuality, the testbed imaging system is structured as a set of programs, communicating through the use of files. These programs utilize independent modules that group logically related functions. Currently defined modules include objects, light sources, materials, texture maps, and a variety of reflection models. For example, each unique geometric entity is a separate object module and is controlled by its object manager. The object manager is responsible for maintaining its own internal data structure, file input and output, intersection calculations of the object with an arbitrary ray, and a number of other tasks specific to the object. Interfaces with all object managers are procedurally defined, making it possible for the global testbed system to operate without specific knowledge of the details within each object manager.

This modular approach has flexibility advantages and operating speed disadvantages. Because of the modularity, an experimental object can easily be added and debugged without affecting the operation of the entire system. Every object is handled identically by the ray-tracing simulation routines, which are performed with a small body of code that is easily understood and maintained. Thus, the testbed imaging system is not affected by an alteration in the number of available object entities, provided the interactive menu control program has sufficient flexibility. But modularity results in a time penalty because each request from the ray-tracing simulation routines must be decoded through a module manager, no matter how simple the request. With our constantly changing research requirements in mind, we felt that this speed trade-off was justified by the flexibility and simplicity inherent in the system.

Image storage. For research comparisons, it was important to store the data describing the image plane intensity function at a high spatial and spectral resolution and to eliminate any constraints imposed by display limitations. The image information gathered by the ray tracer is transformed from the spectrally sampled values used for computation into the CIEXYZ color space immediately before encoding and file writing. (CIEXYZ is a colorimetry system, based on three primaries, established in 1931 by the Commission Internationale de l'Eclairage.) Three 16-bit-deep channels of any desired spatial resolution are used to store the respective XYZ intensity values of the image in run-length encoded format.

The CIEXYZ color space was chosen because it is possible to represent all perceptible colors in the positive octant of the color space and because transformation from this format into any other color display space is easily accomplished. The 16-bit-word-per-channel resolution was selected so that very high color resolution could be retained. These bounds are large enough to assure adequate color resolution when display processing is performed and they remove the constraints of the display or recording color resolution from the intensity function calculations.

In preparing an image for display and/or recording, it is necessary to consider the limitations of the display and/or recording device and adjust the image accordingly. These limitations include spatial resolution, spectral resolution, color gamut (in both displayable chromaticities and intensity range), and nonlinearities of the display and/or recording device. The required corrections for some of these limitations, such as gamma-correcting for nonlinearity, are straightforward. The required adjustments for other limitations, such as the color gamut and dynamic range, involve the perception of the displayed and/or recorded image. These adjustments cannot eliminate errors in the display, but may render them imperceptible.

A matrix can be generated that will transform intensity information from the CIEXYZ space into a color space described by three primary phosphors in the case of monitor display, or into a color space described by primaries representing the composite behavior of the monitor, photographic filters, film sensitivity, and projection media in the case of film recording. This matrix requires knowledge of the chromaticities and maximum intensities of the three primaries. The generation of this transformation matrix is detailed by Meyer,[21] Meyer and Greenberg,[22] Neal,[23] and Wintringham.[24]

Results

The following describes some of the beneficial results of experimentation conducted to date. (Definitions of terms used in the illumination model equations are listed in Table 1.)

Illumination model improvements. Historically, reflection models have considered the illumination function to be composed of two portions—light reflection as a result of direct illumination from light sources, and light reflection and transmission as a result of the global illumination from other parts of the environment. Perhaps the first realistic images were modeled by Phong,[9] whose model included diffuse and specular terms for the direct light sources and a constant global ambient term (Figure 5a):

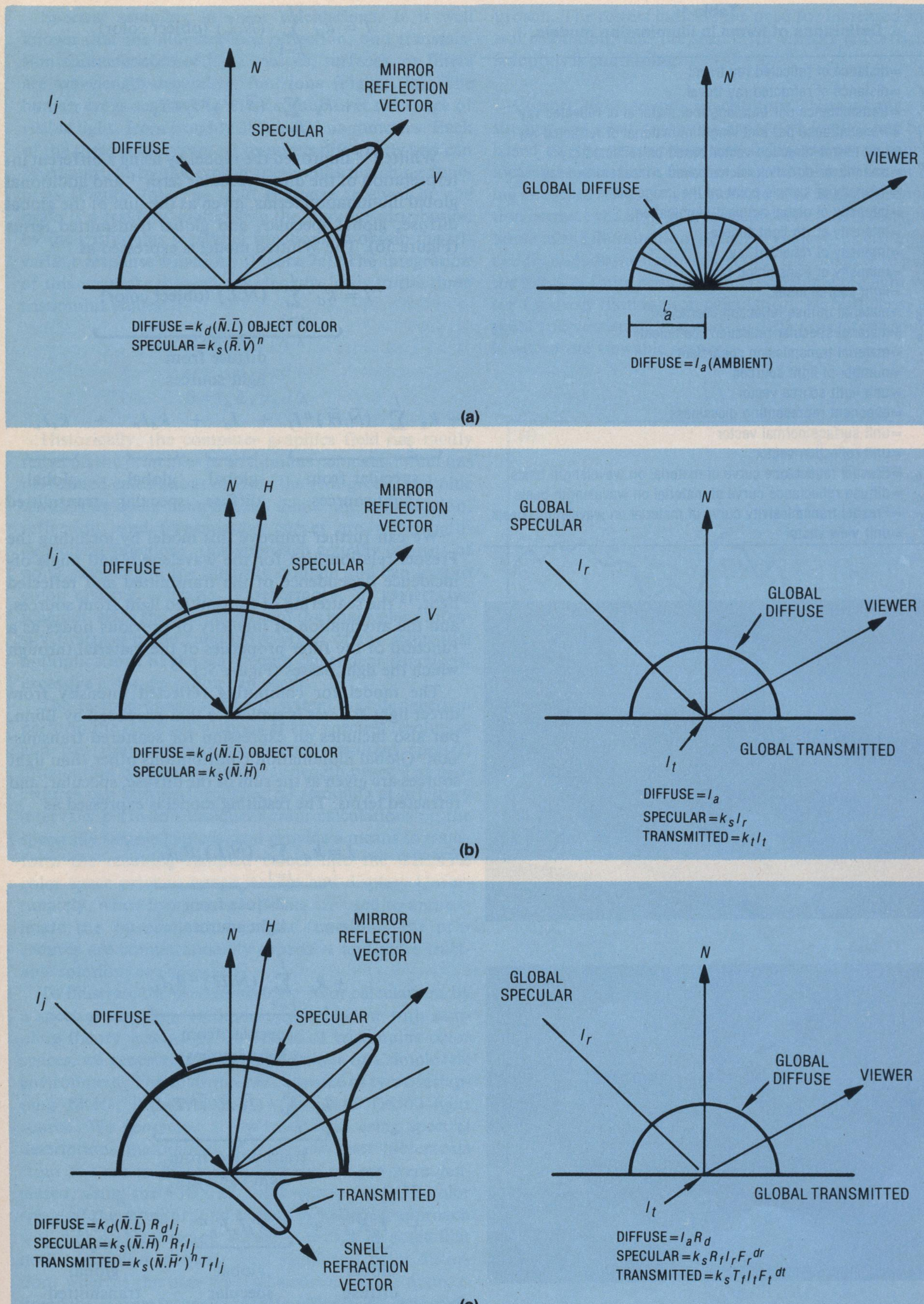


Figure 5. The Phong (a), Whitted (b), and Hall (c) illumination models, with specific light source models at left and global illumination models at right.

Table 1.
Definitions of terms in illumination models.

dr	=distance of reflected ray travel
dt	=distance of refracted ray travel
F_r	=transmittance per unit length of material of reflected ray
F_t	=transmittance per unit length of material of refracted ray
$\bar{H}$	=unit mirror-direction vector based on reflected ray
$\bar{H}'$	=unit mirror-direction vector based on transmitted ray
I	=intensity of sample point on the image plane
I_a	=intensity of global ambient illumination
I_j	=intensity of jth light source
I_r	=intensity of reflected ray
I_t	=intensity of transmitted ray
j	=light source index
k_d	=material diffuse reflection coefficient
k_s	=material specular reflection coefficient
k_t	=material transmission coefficient
l	=number of light sources
$\bar{L}$	=unit light source vector
n	=exponent representing glossiness
$\bar{N}$	=unit surface normal vector
$\bar{R}$	=unit reflection vector
R_f	=Fresnel reflectance curve of material on wavelength basis
R_d	=diffuse reflectance curve of material on wavelength basis
T_f	=Fresnel transmissivity curve of material on wavelength basis
$\bar{V}$	=unit view vector

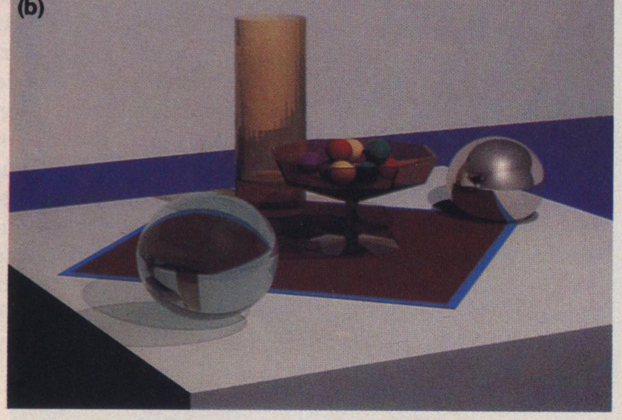


Figure 6. A ray-traced image based on the Whitted interface model (a); the same image based on the Hall improved interface model (b).

$$I = k_d \sum_{j=1}^{l} (\bar{N}.\bar{L}) \text{ (object color)}$$

$$+ k_s \sum_{j=1}^{l} (\bar{R}.\bar{V})^n I_j + I_a.$$

Whitted[19] improved the model by using a different interpretation of the direct specular term[11] and additional global illumination terms, given as the sum of the global, diffuse, global specular, and global transmitted terms (Figure 5b). The Whitted model is expressed as

$$I = \underbrace{k_d \sum_{j=1}^{l} (\bar{N}.\bar{L}) \text{ (object color)}}_{\text{diffuse from light sources}}$$

$$+ \underbrace{k_s \sum_{j=1}^{l} (\bar{N}.\bar{H})^n I_j}_{\text{specular from light sources}} + \underbrace{I_a}_{\text{global diffuse}} + \underbrace{k_s I_r}_{\text{global specular}} + \underbrace{k_t I_t}_{\text{global transmitted}}.$$

We can further improve this model by including the Fresnel relationships for the wavelength and angle-of-incidence dependence of the transmitted and reflected light,[13] the scattering of transmitted light from sources, and the attenuation of intensity of previous nodes as a function of the filter properties of the material through which the light passes (Figure 5c).[25]

The model for computing reflected intensity from direct light sources is similar to that proposed by Blinn, but also includes an expression for scattered transmission. Global illumination contributions other than light sources are given as the sum of the diffuse, specular, and refracted terms. The resulting model is expressed as

$$I = \underbrace{k_d \sum_{j=1}^{l} (\bar{N}.\bar{L}) R_d I_j}_{\text{diffuse from light sources}}$$

$$+ \underbrace{k_s \sum_{j=1}^{l} (\bar{N}.\bar{H})^n R_f I_j}_{\text{specular from light sources}}$$

$$+ \underbrace{k_s \sum_{j=1}^{l} (\bar{N}.\bar{H}')^n T_f I_j}_{\text{transmitted from light sources}}$$

$$+ \underbrace{I_a R_d}_{\text{global diffuse}} + \underbrace{k_s R_f I_r F_r^{\,dr}}_{\text{global specular}} + \underbrace{k_s T_f I_t F_t^{\,dt}}_{\text{global transmitted}}$$

A comparison of ray-traced images based on the Whitted model with images based on the Hall improved interface model is shown in Figure 6.

Spectral sampling in color calculations. It is well known that the illumination, reflection, and transmission characteristics of light sources, surfaces, or filters are wavelength-dependent functions (Figure 7a). The human eye is sensitive to these stimuli over the range of visible light, from roughly 380 to 800 nanometers. Each of the three color receptors responds differently and can be quantified by the typical tristimulus matching functions (Figure 7b). The total response received by the brain is a result of multiplying the stimulus distribution by the tristimulus matching functions to obtain the trivariant response functions (Figure 7c). The integration of this response over all wavelengths leads to the three tristimulus values:

$$R=\int_\lambda E_\lambda \bar{r}_\lambda d\lambda$$
$$G=\int_\lambda E_\lambda \bar{g}_\lambda d\lambda$$
$$B=\int_\lambda E_\lambda \bar{b}_\lambda d\lambda$$

Historically, the computer graphics field has tacitly relied on the principles of tristimulus colorimetry but has performed calculations only on the red, green, and blue tristimulus color components. Since the illumination, reflection, and transmission curves are wavelength-dependent functions, this approach actually consists of point sampling in the intensity domain and can lead to color aliasing artifacts. Furthermore, in ray-tracing schemes, since the computation of the image plane intensity function is the cumulative effect of the sequential multiplications of the wavelength-dependent functions, excessive color distortion can occur.

The testbed image synthesis system was designed to perform the color calculations at any spectral resolution and can thus compute the correct resultant spectral distribution on a continuous wavelength basis. The procedures can also perform spectral sampling at arbitrary intervals, perform subsequent color calculations on the spectrally sampled values, and provide a means to transform the resulting sampled values into the CIEXYZ color space prior to image storage and display. Unfortunately, when many sample points are used to approximate the wavelength-dependent functions, the procedures are computationally expensive and more tractable solutions are necessary.

To illustrate the need to perform color calculations by a spectral sampling methodology consistent with sampling theory instead of by means of tristimulus color spaces, we generated comparison images of a simple test environment. The environment consisted of two overlapping filters, illuminated by a standard D6500 light source. We generated a control image, using spectral descriptions maintained at one-nanometer increments from 360 nm to 830 nm. Additional images were generated, using the CIEXYZ color space, the RGB color space of the monitor, and a spectral sampling approach with nine spectral sample values. It should be noted that both tristimulus methods yield significant color distortion, whereas the nine spectral samples closely approximate the correct intensity function (Figure 8). Although the intensity calculation time increased, the total computation time for the nine-sample-point approach was less than two percent greater than the tristimulus approach. The results indicate the need for increased spectral resolution, and the search for correct and efficient solutions is continuing.

Correct light source descriptions. Historically, the shading for most computer-generated images has been based on the assumption that lights are point sources located at an infinite distance from the environment being rendered. This assumption reduces image computation because the light source vector direction is constant. Some of the illumination models previously discussed explicitly include the light vector; thus it is possible to place the lights at finite positions and recalculate the light vector for every illumination calculation. Nevertheless, the point source assumption leads to difficulties if the source is within the viewable scenes.

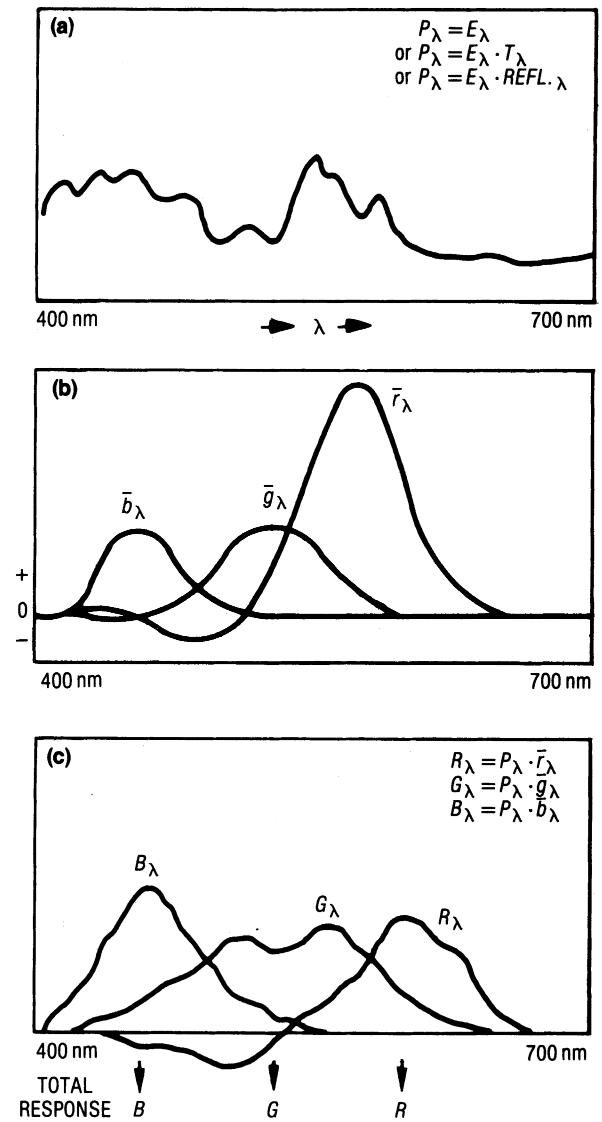


Figure 7. The response of a trivariant system to a stimulus: (a) stimulus distribution function $P\lambda$; (b) tristimulus matching functions $\bar{r}_\lambda$ $\bar{g}_\lambda$ $\bar{b}_\lambda$; (c) trivariant response functions R_λ G_λ B_λ.

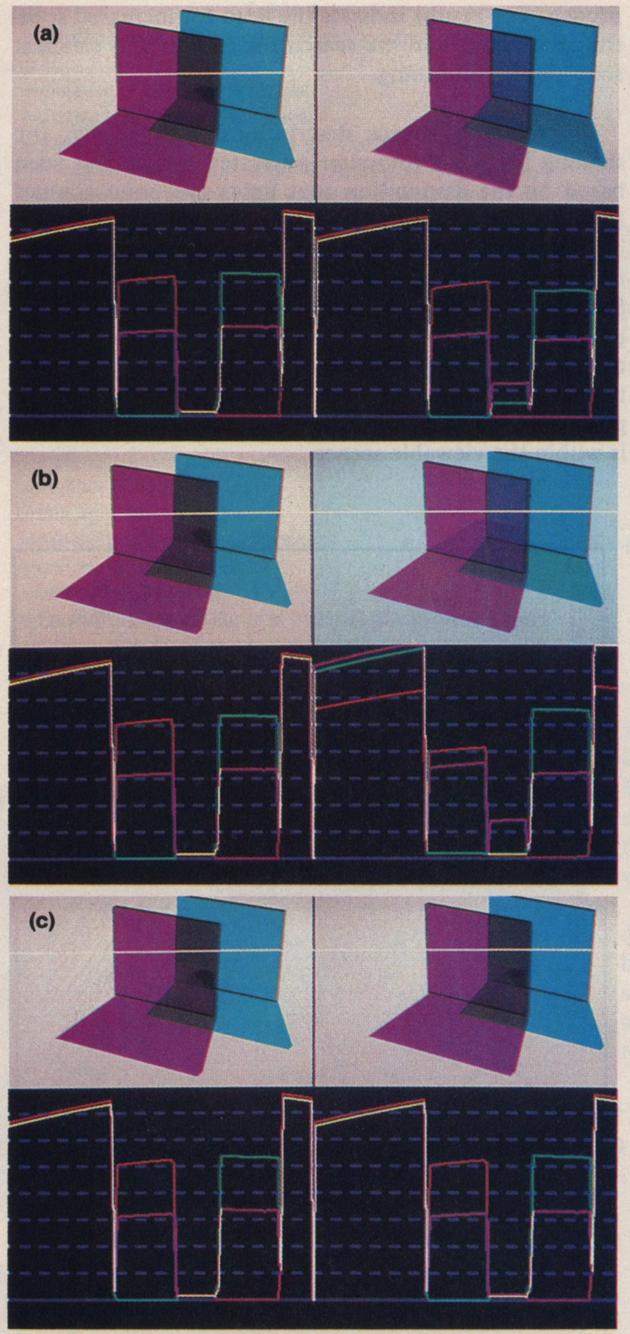


Figure 8. Comparison of spectral sampling techniques: three samples in CIEXYZ color space (a); three samples in RGB color space (b); nine spectral samples (c). The RGB intensity graphs are shown for the same scan line in each image. The control image, computed on a wavelength basis, appears on the left of each test image.

Table 2.
Comparative statistics for adaptive tree-depth control in complex image of mirrored room.

MAXIMUM TREE DEPTH	ADAPTIVE CONTROL DEPTH	AVERAGE TREE DEPTH	IMAGE GENERATION TIME
1	NO	1.0	264 MIN.
15	NO	14.89	3941 MIN.
15	YES	1.71	480 MIN.

Luminaire data must include the geometric descriptions, positional information, and the spatial and spectral intensity information. The testbed image synthesis system has incorporated this information. The geometry of the luminaire is completely defined as either a point source, surface of revolution, or a linear light source such as a fluorescent light fixture.[20] The position and orientation of each fixture is defined in the environment builder. Each light source also has two attributes associated with it. The spatial intensity information is described by goniometric diagrams of revolution, such as Figure 9.[26] For nonsymmetric light sources, the vector intensity information can be interpolated between two goniometric diagrams associated with each of the principal axes of the luminaire. This method for describing the intensity distribution of a light source is typical in the lighting industry. Each light source also has a spectral intensity distribution, defined in the same manner as the material reflectance curves.

With this information, it is possible to generate images that not only contain the lights, but accurately simulate the light distribution. The lights depicted in Figure 10 and in the image shown on the cover were modeled as surfaces of revolution, with an intensity distribution defined by a cosine function. This intensity function is similar to that used for photographic light simulation, recently presented by Warn.[27] The intensity contribution of the light source for each ray was determined by use of the goniometric diagram of the source.

Adaptive tree-depth control. Ray tracing combines the visible surface determination with a means for accumulating the global illumination information. Traditionally, an intersection tree for any sample point is constructed to an arbitrary depth prespecified to assure that relevant reflections and refractions are captured in the image. For most typical environments, however, the percentage of the image that is comprised of reflective or transparent surfaces is relatively low. Thus, there is considerable potential for computational savings if the calculations are carried out only when necessary.

Consider, for example, the image of a mirrored room in Figure 10a. Although it appears to have vast reflecting areas, only a small percentage of the image actually consists of reflective surfaces. However, the specular reflections are very important in generating the correct representation of the environment and must be traced to a depth of 15 reflections to recover sufficient image information. For comparison, Figure 10b shows the same image generated without reflection—that is, to a tree depth of one. Obviously, the resulting image is less than adequate.

We can approximate the upper bound of the contribution of any node of the intersection tree to the final color of the sample point, by considering the intensity of every node on the tree to be maximum intensity. The first node on the tree will contribute 100 percent to the final color of the sample point. The maximum contribution of the children nodes can be evaluated by setting I_r and F_r to unity for the reflected child and I_t and F_t to unity for the transmitted child and then computing the respective contributions. This process can be repeated for the children of any parent node in the tree. By multiplying the approximate maximum contributions, we can compute the maximum cumulative contribution of the child node to the sample point. Thus, by establishing a cutoff contribution threshold, we can adaptively control the depth of the tree during the ray-tracing process.

The potential of this adaptive tree-depth control for improving the computational efficiency in ray tracing was tested on several environments. Statistics for the generation of the complex gallery environment of Figures 10a and 10b are given in Table 2. Significantly, the average tree depth, even for this very reflective environment, was 1.71, proving the efficiency of the approach.

As we have shown here, Cornell University's testbed image synthesis system has helped in the development of a number of improvements of existing ray-tracing techniques. In summary, these include a hybrid reflection model, a spectral-sampling method, the incorporation of light sources, and adaptive tree-depth control. Nevertheless, we point out again that many of the problems of realistic image generation through ray tracing have not been solved. The inherent point-sampling technique is prone to aliasing, the reflection models proposed do not include the correct energy formulations, and the computational expense is currently too high. The use of environmental image coherence techniques would probably make these approaches more tractable. It is our hope that the testbed imaging system will provide an appropriate environment for further exploration of these problems. ■

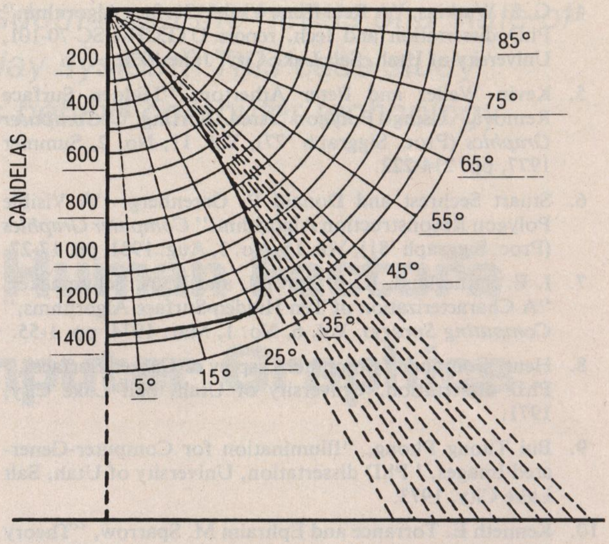
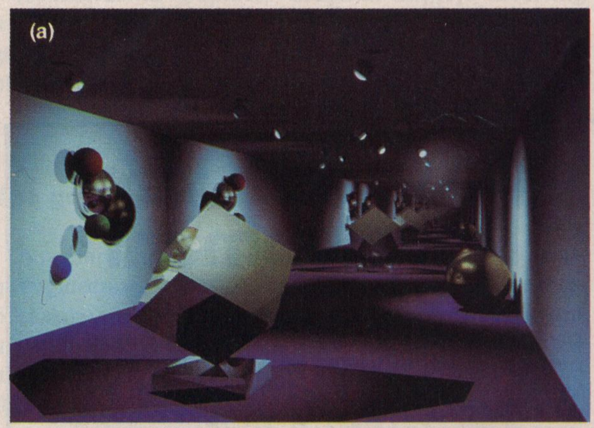


Figure 9. Goniometric diagram of luninaire showing intensity variations with vector direction.



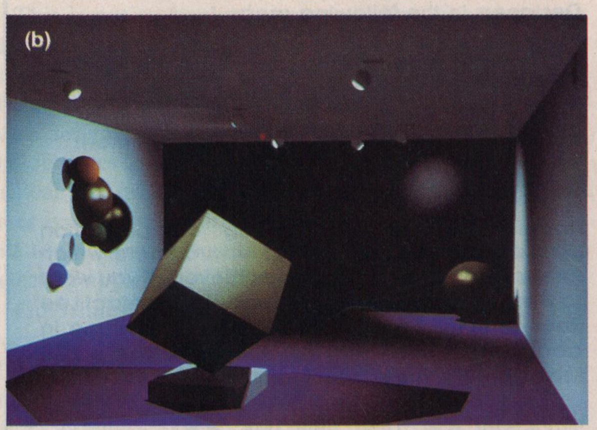


Figure 10. Images of gallery with mirrored walls and local light sources: adaptive tree depth (a); maximum tree depth = 1 (b).

Acknowledgments

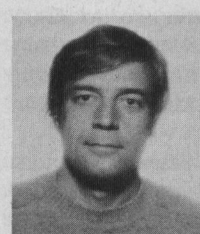
Many people from the Program of Computer Graphics contributed to this research and the creation of the testbed imaging system. Special thanks go to Gary Meyer, Ted Crane, Chan Verbeck, Rikk Carey, Gary Hooper, Hank Weghorst, and Emil Ghinger. We hope that this testbed environment will provide a foundation for their continued research.

We are also grateful to the National Science Foundation, whose support was instrumental in the initiation of the laboratory and who directly supported this research under grant no. MCS-8203979, "Interactive Computer Graphics Input and Display Techniques."

References

1. R. Schumacker, B. Brand, M. Gilliland, and W. Sharp, "Study for Applying Computer Generated Images to Visual Simulation," AFHRL-TR-69-14, US Air Force Human Resources Laboratory, 1969.
2. J. Newell, R. Newell, and T. Sancha, "A Solution to the Hidden Surface Problem," *Proc. ACM Nat'l Conf.*, 1972, pp. 443-450.
3. John E. Warnock, "A Hidden Surface Algorithm for Computer Generated Halftone Picture Representation," tech. report 4-15, University of Utah, Salt Lake City, June 1969.

4. G. S. Watkins, "A Real-Time Visible Surface Algorithm," PhD dissertation and tech. report UTECH-CSC-70-101, University of Utah, Salt Lake City, June 1970.
5. Kevin Weiler and Peter Atherton, "Hidden Surface Removal Using Polygon Area Sorting," *Computer Graphics* (Proc. Siggraph '77), Vol. 11, No. 2, Summer 1977, pp. 214-222.
6. Stuart Sechrest and Donald P. Greenberg, "A Visible Polygon Reconstruction Algorithm," *Computer Graphics* (Proc. Siggraph '81), Vol. 15, No. 3, Aug. 1981, pp. 17-27.
7. I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys,* Vol. 6, No. 1, Mar. 1974, pp. 1-55.
8. Henri Gouraud, "Computer Display of Curved Surfaces," PhD dissertation, University of Utah, Salt Lake City, 1971.
9. Bui Tuong Phong, "Illumination for Computer-Generated Images," PhD dissertation, University of Utah, Salt Lake City, 1973.
10. Kenneth E. Torrance and Ephraim M. Sparrow, "Theory for Off-Specular Reflection from Roughened Surfaces," *J. Optical Society of America,* Vol. 57, No. 9, Sept. 1967, pp. 1105-1114.
11. James F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics* (*Proc. Siggraph '77*), Vol. 11, No. 2, Summer 1977, pp. 192-198.
12. James F. Blinn, "Computer Display of Curved Surfaces," PhD dissertation, University of Utah, Salt Lake City, 1978.
13. Robert L. Cook and Kenneth E. Torrance, "A Reflection Model for Computer Graphics," *ACM Trans. Graphics,* Vol. 1, No. 1, 1982, pp. 7-24.
14. Peter Beckmann and Andre Spizzichino, *The Scattering of Electromagenetic Waves from Rough Surfaces,* MacMillan, New York, 1963, pp. 1-33, 70-98.
15. James F. Blinn and M. E. Newell, "Texture and Reflection in Computer Generated Images," *Comm. ACM,* Vol. 19, No. 10, 1976, pp. 542-547.
16. A. Appel, "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," *Proc. ACM Nat'l Conf.*, 1967, pp. 387-393.
17. R. A. Goldstein and R. Nagel, "3-D Visual Simulation," *Simulation,* Jan. 1971, pp. 25-31.
18. Douglas Scott Kay, "A Transparency Refraction and Ray Tracing for Computer Synthesized Images," master's thesis, Cornell University, 1979.
19. Turner Whitted, "An Improved Illumination Model for Shaded Display," *Comm. ACM,* Vol. 23, No. 6, June 1980, pp. 343-349.
20. Channing P. Verbeck, Untitled, unpublished master's thesis, Cornell University, 1984.
21. Gary W. Meyer, "Colorimetry and Computer Graphics," Cornell University, Program of Computer Graphics Report 83-1, Apr. 1983.
22. Gary W. Meyer and Donald P. Greenberg, "Perceptual Color Spaces for Computer Graphics," *Computer Graphics* (Proc. Siggraph '80), Vol. 14, No. 3, Aug. 1980, pp. 254-261.
23. C. B. Neal, "Television Colorimetry for Receiver Engineers," *IEEE Trans. Broadcast Television and Receiver,* Aug. 1973, pp. 63-73.
24. W. T. Wintringham, "Color Television and Colorimetry," *Proc. IRE,* Oct. 1951, pp. 1135-1172.
25. Roy A. Hall, "A Methodology for Realistic Image Syntehsis," master's thesis, Cornell University, 1983.
26. R. Christopher Odgers, "An Interactive Computer Aided System for Artificial Lighting Design," thesis, Cornell University, 1978.
27. David R. Warn, "Lighting Controls for Synthetic Images," *Computer Graphics* (Proc. Siggraph '83), Vol. 17, No. 3, July 1983, pp. 13-21.



Roy Hall develops image generation software at Robert Abel & Associates. His current focus is increased image realism within the constraints of a film production environment.

Hall received bachelor's degrees in civil engineering and architecture from Rensselaer Polytechnic Institute in 1976. For the next several years, he was a structural engineer for Birdair Structures, where he was introduced to computer graphics. He received an MS in computer graphics from Cornell University in 1983.

Donald P. Greenberg is director of the Program of Computer Graphics and the Computer-Aided Design Instructional Facility at Cornell University. He presently teaches in the departments of Computer Science and Architecture. Researching and teaching in the field of computer graphics since 1966, Greenberg's specialties include hidden-surface algorithms, geometric modeling, color science, and synthetic image generation.

Greenberg received his PhD in structural engineering from Cornell University. He is a member of the IEEE and the ACM and has published more than fifty articles related to computer graphics and computer-aided design.