Multi-Pass Pipeline Rendering: Realism For Dynamic Environments

Paul J. Diefenbach*and Norman I. Badler†

Center for Human Modeling and Simulation, University of Pennsylvania, Philadelphia PA 19104-6389

Abstract

A coordinated use of hardware-provided bitplanes and rendering pipelines can create fast ray traced quality illumination effects for dynamic environments by using Multi-pass Pipeline Rendering (MPR) techniques. We provide recursive reflections and refractions through the use of secondary viewpoints and projective image mapping. We extend the traditional use of shadow volumes to provide global direct illumination effects which fit into our recursive viewpoint paradigm. Hardware surface shading is fit to a physically-based BRDF to provide a better local model, and the framework permits incorporation of indirect illumination as well. Furthermore, material transmittance is approximated using an extension to projective textures. Together, these techniques provide a platform for producing realistic images in highly dynamic environments. While most appropriate for scenes which specular components contribute largely to the secondary illumination, the integration of MPR with indirect radiosity solutions also provides a dynamic solution for highly diffuse environments. These techniques are immediately applicable to areas such as walkthroughs, animation, and interactive dynamic environments to produce more realistic images in near real-time.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Bitmap and framebuffer operations; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Color, shading, shadowing, and texture.

*Currently at DRaW Computing Associates Inc., 3508 Market St., Philadelphia, PA 19104. E-mail: pjdief@drawcomp.com

†E-mail: badler@graphics.cis.upenn.edu

1997 Symposium on Interactive 3D Graphics, Providence RI USA

1 INTRODUCTION

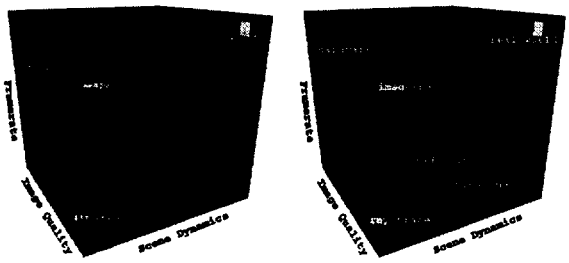
With large investments in sophisticated (and costly) graphics hardware, why is so much rendering performed off-line? While hardware-based rendering pipeline systems such as the SGI architecture do permit real-time interaction, the quality of the images they produce has been limited. In contrast, ray tracing systems produce very accurate scenes of specular environments, but each image requires significant computation time. Radiosity systems provide accurate object-space diffuse lighting representations, but only with a large precomputation overhead and for relatively static environments. Traditionally, to produce an image such as Fig. 1, one of these two methodologies has been used. This image, however, was produced using only hardware-provided graphics and pipeline rendering techniques.

Figure 1: Multi-pass Pipeline Rendered Image (1 overhead and 4 vanity lights)

Multi-pass Pipeline Rendering (MPR) extends a series of related multi-pass techniques analogous to Heckbert and Hanrahan's beam tracing [19] to provide added image quality for interactive, dynamic environments. Because it relies

on multiple rendering passes to add additional levels of detail in each pass, it can easily be tailored to suit the user's desired performance-quality needs. Graphically, the relationship between performance, dynamics, and quality for traditional systems is seen in Figure 2(a). MPR provides a broad spectrum of image quality and realism which fits into our performance model in Fig. 1(b).

Current hardware rendering has been limited in the effects it provides. High polygonal rendering rates and texture mapping provide added detail, but scene illumination has gone relatively unchanged. Shadows have been somewhat supported through shadow volumes [8] and shadow buffers[30], and some specular reflective surfaces have been modeled using texturing[32][4] and secondary viewpoints[23]. There has been, however, almost no concerted effort to use the hardware graphics to achieve physics-based rendering. Multi-pass Pipeline Rendering begins to address this issue.



(a) Traditional (a) MPR

Figure 2: Traditional vs MPR Relationship

In MPR, reflective and refractive images are provided through secondary renderings from virtual viewpoints. In addition, all facets of the illumination of the scene for the primary and secondary images are also accomplished through multiple hardware-based renderings. We extend the traditional use of shadow volumes to provide global direct illumination effects which fit into our recursive viewpoint paradigm. Hardware surface shading is fit to a physically-based BRDF to provide a better local model, and the framework permits incorporation of indirect illumination as well. Furthermore, material transmittance is approximated using an extension to projective textures. Together, these techniques provide a platform for producing realistic images in highly dynamic environments. In addition, the multi-pass architecture presented readily extends to a progressive refinement approach for interactive rendering. These techniques are immediately applicable to areas such as walkthroughs, animation, and interactive dynamic environments to produce more realistic images in near real-time.

2 DEFINITIONS

For the purposes of this paper, we shall introduce terms common to users in the GL environment. *Stencil planes* are an enhanced Z-buffer mentioned in [5]. In its simplest form, pixels are written only if the current stencil value (analogous to the current Z value) of the destination pixel passes the defined stencil test. Depending on the result, the pixel is written and the stencil value is changed.

Shadow volumes are volumes bounded by silhouette faces. A silhouette face is a face created for each edge of an object by extending that edge away from the light source along the light-ray direction.

An *accumulation buffer* is a secondary image buffer to which the current image can be added. The resulting image can also be divided by a constant. This enables a blending of images or image features.

In-out refractions are refractions which occur when light passes from one medium to another and back to the first, such as light traversing through a piece of glass. There is an entry refraction and an exit refraction, producing a refracted ray parallel to the incident ray.

3 SPECULAR IMAGES

Reflections and refractive transparency are important for lending realism to many scenes, in particular, interior environments. A specular image corresponds to an image from a secondary viewpoint mapped onto the specular surface. We term a specular surface to include both specular reflective and specular transmissive (refractive) surfaces as used in Wallace *et. al.*'s definition of light transport[33]. For example, a specular reflected image is the flipped image from a viewpoint on the "other" side of the mirror. This analogy provides the basis for mirror reflection in several systems[23][19].

In the MPR method, specular images are implemented by rendering the entire environment, exclusive of the specular surface. The specular surface is drawn with Z-buffering, creating a stencil mask on pixels where the mirror is visible. A second rendering of the environment is then performed from the virtual reflected or refracted viewpoint, drawing only over the previously masked pixels. This virtual viewpoint can be determined using the beam-tracing specular transformations[19]. For specular refractive surfaces, the refractive image must be further transformed to "fit" onto the refractive surface. Recursion is provided through maintaining the depth information within the stencil mask values. This method is fully described in [10]. For curved surfaces, tessellation or environment mapping[4] can be used. Another promising area is the use of field-of-view adjustment for certain classes of curved surfaces.

In addition, translucent and other light dispersing materials can be simulated using the hardware fog feature and the stencil planes. Translucent objects act as a filter with closer objects more clearly visible than farther objects due to the random refractions which take place[22]. This effect can be approximated using hardware fog features with the minimum fog set at the refractive plane distance and the maximum at the desired distance depending on the material property. Although fog is linear with respect to the view, the approximation is fairly accurate due to the limited angular displacement of the refracting plane because of the critical angle. A more versatile fog function supporting a general linear transform would provide much more accurate translucency and light scattering for reflective surfaces. By incorporating multisampled stochastic (X, Y) shearing about the specular surfaces normal axis, light scattering through the translucent material can also be simulated. This is accomplished by accumulating- using an accumulation buffer[16]- intermediate stochastically-sheared specular images to produce the final scattered effect.

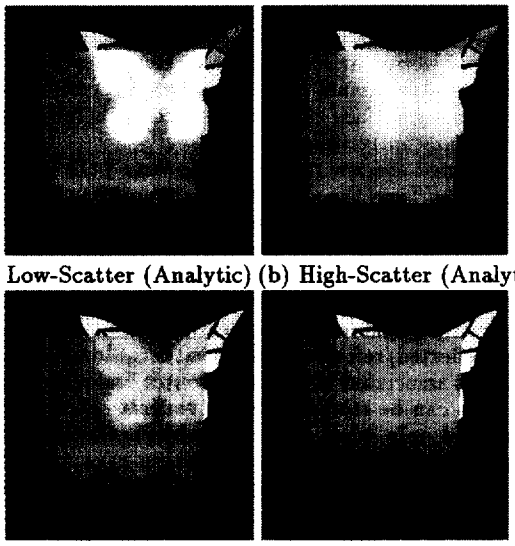
A 4x4 transform is created which includes a stochasti-

cally generated (X, Y) shear. This transform is premultiplied by the global inverse transform of the specular face normal and postmultiplied by the global transform of the normal, with the resulting transform pushed onto the view matrix stack. This creates a shear linear with respect to the perpendicular distance from the specular surface.

An overview of the entire rendering process is seen in the following code.

```
mask_face(spec_face);       //set stencil area for face
reclassify_camera(Camera); //move to virtual viewpoint
enable_clip_plane(spec_face); //clip away geometry
for (i=0; i++; i<numsamples){  //loop over all samples
   // -shear view stochastically around specular normal
   shear_view(Camera,spec_face,sample[i]);
   draw_window(Camera);  // -recursively draw
   accbuf(ACCUMULATE);   // -add intermediate images
}
disable_clip_plane(spec_face);//turn off clipping plane
accbuf(RETURN);           //display composite image
```

By adjusting the jitter amount and the fog parameters, this method can provide a range of effects similar to those produced by analytic methods[2], although at a much lower cost. Figure 3 compares analytically-generated images[1] (a) and (b) with our multi-pass images (c) and (d). Our frosted glass images were each generated in less than 0.1 seconds, as compared to 6.5 seconds in the analytic approach. Figure 4 compares the two approaches for a scattering reflective surface. Here, analytically-computed image (a) required 1-2 minutes of rendering time, as compared to under 0.5 seconds for multi-pass images (b) and (c).



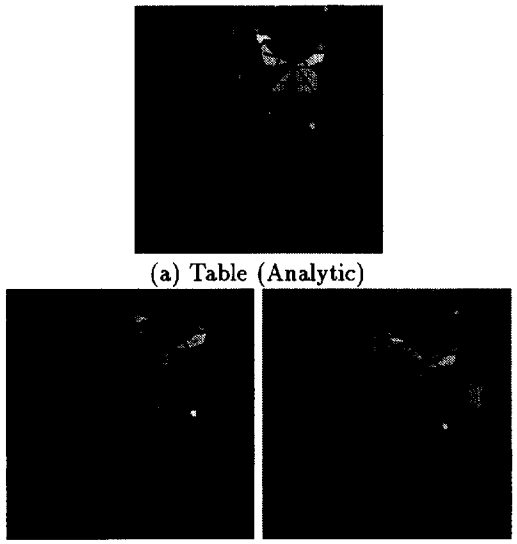
(a) Low-Scatter (Analytic) (b) High-Scatter (Analytic)

(c) Low-Scatter (MPR) (d) High-Scatter (MPR)

Figure 3: Frosted Glass

Figure 3 demonstrates the dispersing nature of a translucent scattering surface under varying shear multipliers (m) and fog parameters (fmax). Note how the elongated blue beam is clearer the closer it is to the glass. Figure 4 again demonstrates this effect in conjunction with texture mapping to simulate a shiny marble surface. As this method uses

[1] Butterfly design by Elsa Schmid[29].

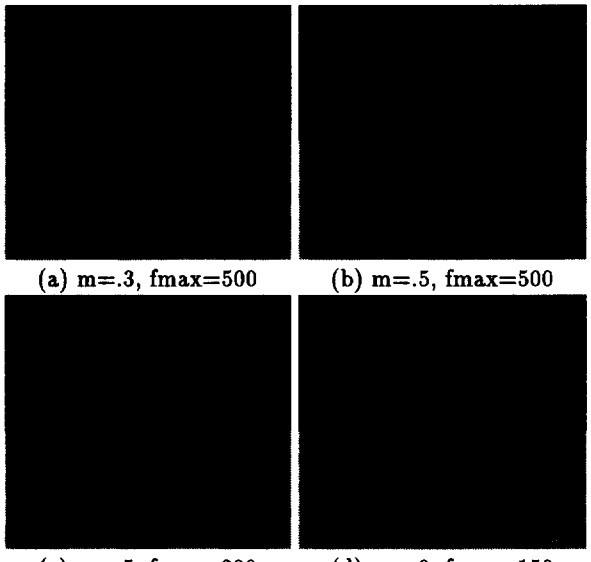


(a) Table (Analytic)

(b) Table (MPR) (c) Wood (MPR)

Figure 4: Glossy Table

hardware-based rendering, image size has minimal effect on timings in contrast to ray tracing. In systems where rasterization is independent of polygon size (i.e. Pixel-Planes[15]), image size is not a consideration.



(a) m=.3, fmax=500 (b) m=.5, fmax=500

(c) m=.5, fmax=300 (d) m=.9, fmax=150

Figure 3. Frosted Glass

4 GLOBAL DIRECT ILLUMINATION

Whereas the original implementation of multiple-pass techniques presented in [10] demonstrated how shadow and light volume generation can work with specular image stenciling to produce global direct illumination effects, it suffered from several shortcomings. These included low performance due to costly intersection checks and unneeded shadow volume recomputation, as well as incorrect multi-source blending.

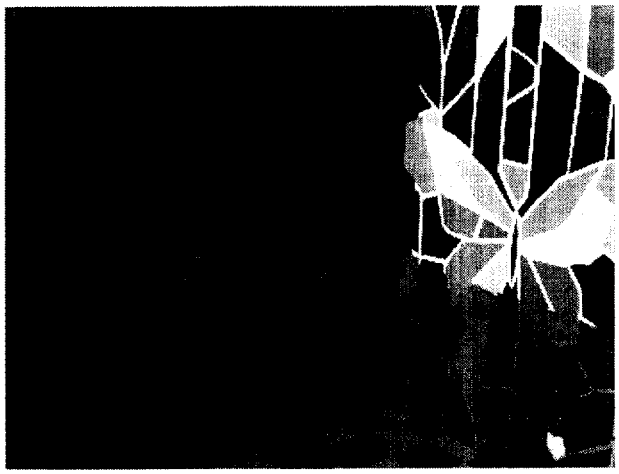


Figure 4. Glossy Marble Wall

These issues are addressed in the MPR implementation described here.

4.1 Shadows

In MPR, shadows are implemented based on Brotman and Badler's [5] extension of Crow's shadow volume method[8]. This technique uses the plus-minus principle of silhouette faces to mask regions inside the shadow volume.

The use of shadow volumes is more suitable to our application than the shadow buffer[28] method due to several factors. These include the limited "field of view" of shadow buffers as well as limited resolution. Finally, the most serious drawback for a dynamic environment is that movement of one object requires recalculation of all of the shadow buffer images.

Where the shadow buffer method suffers from the above limitations, the shadow volume method obviates them. The shadow volume is an omni-directional method casting shadows in every direction from a light source. Shadow resolution is at view-space resolution resulting in pixel (or subpixel) accuracy. Object shadows are (mostly) independent which permits recalculation of shadows from only objects which have moved.

4.2 Light Volumes

Light volumes from secondary sources are generated in the same manner as the shadow volumes. Just as shadow volumes stencil the area in shadow, light volumes stencil the area in which light from secondary sources can be added. The light volume is generated using the same plus-minus shadow volume method, however the stencil values are in essence inverted to permit rendering only in the "shadowed" area. The zero value therefore determines the valid render area within a light volume; recursive support uses the methods described in [10].

With each specular-specular transport[33], the light volume is bent by each specular face with which it interacts, and therefore the corresponding virtual light position is transformed as well. Furthermore, the light volume is reduced by each of the specular faces.

As demonstrated in Fig. 5, a light volume from a tertiary source should be clipped by both refractive faces through which it passes. This can be accomplished by either direct

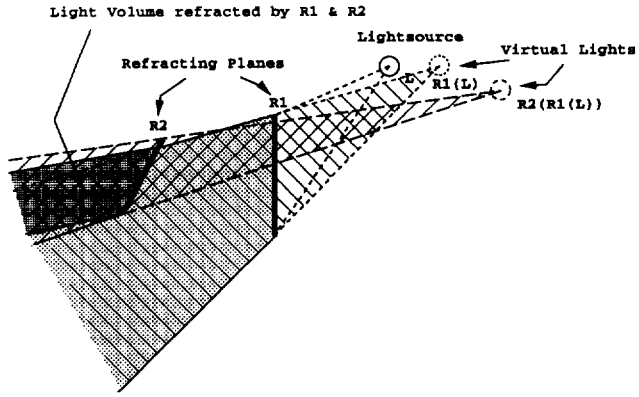


Figure 5: Light volume clipped by both refracting planes.

reduction of the light volume through intersection checks, or by multiple renderings of the light volume through each face to determine the valid overlapping region.

The second method is based on additional passes of the current implementation. As the correct light volume is the light which passes through each specular face along its path, this volume is also the intersection of the light volumes generated through each specular face. Therefore, the correct light volume can be created by stenciling each of the constituent light volumes to determine the overlapping area. While this method is exact for reflective surfaces, it is a close approximation for refractive surfaces based on paraxial rays [19].

4.3 Specular Shadow Volumes

Whereas the method described in [10] traced shadow rays to specular objects, our MPR method computes shadow rays for all specular virtual light sources. This eliminates the need for costly surface intersection checks, thereby rendering images significantly faster.

A reflected shadow ray traces the same path as a shadow ray produced from the associated light source's reflected position. For refractive surfaces, this virtual light source position approximates the refracted shadow ray direction for paraxial rays. Therefore, instead of recursively performing intersection checks for each shadow ray, an approximate solution can be achieved by generating shadow rays for all virtual reflected/refracted light positions. In addition, since each ray is associated with a light source and a specular face, these rays can be stored with the generating object and be regenerated only when either the object, the specular face, or the light source has moved. These rays are stored as shadow volume faces generated for each silhouette edge of the object.

During each image pass, which includes the camera view as well as any reflected or refracted views, shadow volumes are stenciled for the entire environment. In addition, this process is recursively called for each specular face in the environment. This generates shadows and incrementally adds lighting from secondary sources.

While storing shadow volumes significantly reduces rendering times, it can require significant amounts of memory. To reduce this, intelligent creation and re-use of shadow volumes is performed. For our bathroom environment with four light sources at one level of recursion and two specular

faces, 53 megabytes of shadow storage would have normally be needed if volumes were generated for every polygon. Instead, this is reduced in MPR by an order of magnitude through use of intelligent silhouette edge volume generation and re-use of volumes.

In order to understand the need for shadow volume re-use, examine the exponential nature of shadow generation when specular transport is involved. Consider Fig. 6 in which two specular reflective surfaces are involved. To accurately render the lighting effects emitting from mirror R_2 at two levels of recursion, it can be seen that object O produces two shadow volumes involving specular transport from mirror R_1 only, and three shadow volumes involving specular transport from mirror R_1 to mirror R_2. Each additional specular surface or level of recursion requires an exponential increase because it introduces additional shadows at each level of recursion.

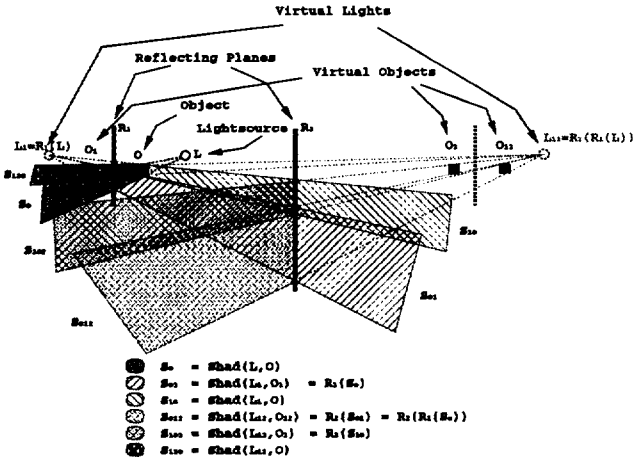


Figure 6: Shadow volumes generated involving R_1.

While it first seems that each of these shadow volumes must be independently stored, this is not necessary and indeed proved too costly. Instead, it can be noted that shadows which pass through a specular surface are simply a transformed version of the original shadow, and therefore a transformed version of the original shadow volume is valid. Again referring to Fig. 6, it can be seen that shadow volumes S_{O12} and S_{1O2} are reflected versions of S_{O1} and S_{1O} respectively, the former itself a reflected version of the shadow directly from the light S_O.

4.4 Light Accumulation

To produce the penumbra effects for areas only in partial shadow, an accumulation of the lighting effects from the primary as well as reflected/refracted lights must occur. There are two methods for performing this accumulation of lighting effects. The first method is to treat each shadow calculation as being independent and to sum each resulting image. The second method uses an extension of shadow volumes for soft shadows[5].

Soft Shadows

Whereas antialiasing is performed by jittering the camera position, soft shadows are created by jittering the light source. This in effect approximates an area light source with a stochastic set of point light sources.

The actual implementation of this source jittering is an approximation to an actual movement of the source point. Recall that shadow volumes are computed once and stored with the corresponding object. Therefore, modification of the light position would require recomputing every shadow volume face for each silhouette edge at each jitter iteration or initial storage of the entire set of jittered faces. Neither of these options is an attractive proposition for an interactive system.

The alternative solution notes that with a jittered source, only two of the shadow volume face's four vertices are moved, and that their movement is a linear transformation of the stochastic jittering of the source. As the distribution is centered around the point source position, any rotation of this distribution also produces a valid stochastic sample.

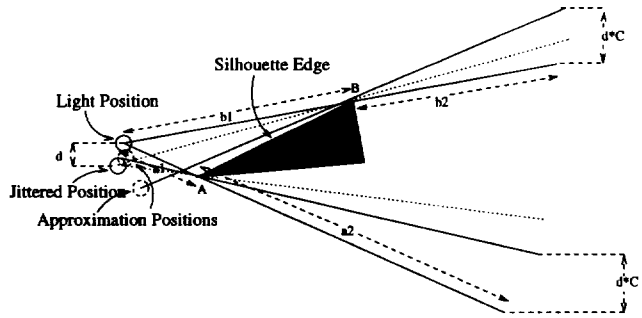


Figure 7: Jitter Approximation Shadow Volumes

As can be seen in Fig. 7, a movement of the light produces a linearly scaled movement of the end of the shadow volume. Fig. 7 demonstrates the above approximation, along with the error introduced by this method. As the segment acts as a pivot, the ratios $b2/b1; a2/a1$ relate the displacement of the two generated nodes of the shadow face to the two nodes bordering on the generating silhouette edge itself. In practice, the jitter size as compared to the distances involved is relatively minute, and a constant multiplier usually suffices.

The above jitter approximation therefore provides a simple way to jitter the shadow volume without its recomputation. A scalar multiplier can be used to control the light source area, and the shape can be directed by the jitter sample distribution itself.

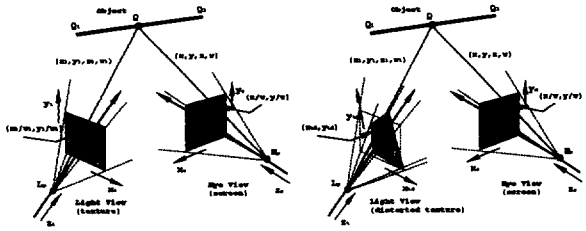
4.5 Specular Projection

The previously described light volumes created for light passing through transparent materials or reflecting off mirrored surfaces have relied on specular surfaces of uniform coloration, density, and transparency. There are, however, many common materials which do not fall into this category, including beveled glass, prisms, and stained glass. Light interaction with these materials can, however, be simulated using projective texture mapping.

Projective textures have been previously used for a variety of purposes. This includes use as a slide projector onto a surface [11] and as an a shadow caster based on a depth texture map [40]. Dorsey[11] used this principle to produce the perspectively distorted image itself. Segal *et.*

al.[30] demonstrated how projective coordinate transformations permit texture coordinate assignment based on the depth maps described in [28]. With this facility available, we can simulate light interaction with many of the above mentioned materials.

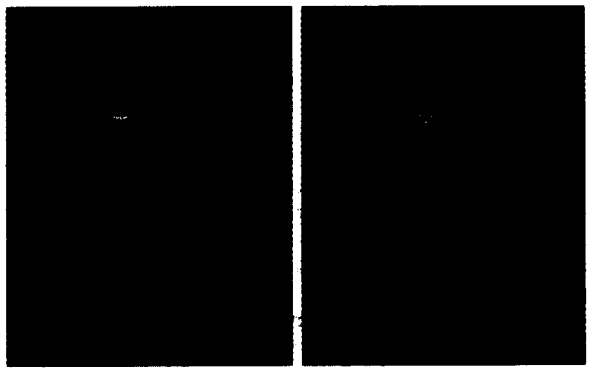
As seen in Fig. 8(a), the texture transformation is dependent on local axial-aligned 2-D coordinate frames. Unfortunately, this is seldom the case with arbitrary environments where the texture plane (i.e. the specular surface) is not axial aligned with the light view. This is demonstrated in Fig. 8(b).



(a) Axial aligned (b) Non-axial aligned

Figure 8: Light and view coordinate systems.

There is however a corresponding projection M_{33} of the texture plane onto the axial aligned texture coordinates. As with the image mapping of a refractive image onto a specular plane[10], this transformation is a 2-D projection of quadrilateral to quadrilateral. In fact, this instance is the simpler mapping of rectangle to quadrilateral, described in [18].



(a) Axial aligned (b) Non-axial aligned

Figure 9: Projected Textures

The application of this transformation is seen in Fig. 9, where the first image shows normal hardware-based texture projection and the second shows the same projection with a non-axial aligned texture plane.

By applying the texture only during the light volume illumination stage, the light volume is thereby modulated by the texture. A detail of these effects is seen in Fig. 10, where four real lights and four virtual sources shining through a "beveled" glass door produce eight partially overlapping texture patterns which are occluded by the door handle shadows.

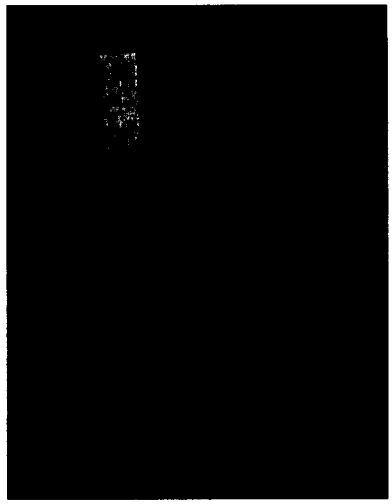


Figure 10: Projected Texture Light Pattern

5 LOCAL DIRECT ILLUMINATION

In the previous section we demonstrated techniques for overlaying and blending multiple renderings to create more realistic images. We have not yet mentioned an equally important consideration, namely the local illumination model. The local model represents the surface shading from direct illumination from real and virtual light sources. These lighting contributions are generated using the available graphics hardware shading model. This generally is Gouraud shading using the Phong lighting model[26][2], which is widely known for its inaccuracies.

There has been much study devoted to both realistic and physically accurate lighting models. This work is generally centered around the rendering equation introduced by Kajiya[20]. This equation, expressed in terms of reflectance

$$L_r(\theta_r,\phi_r) = \int_0^{2\pi}\int_0^{\pi/2} L_i(\theta_i,\phi_i)\,\rho_{bd}(\theta_i,\phi_i;\theta_r,\phi_r)\cos(\theta_i)\sin(\theta_i)\,d\theta_i\,d\phi_i \quad (1)$$

states the relationship between incident light and reflected light from a surface based on a ***bidirectional reflectance distribution function*** (BRDF)[25]. Many BRDF formulations have been developed to try to represent this complex, high-order function. This includes creation of theoretical physics-based models [17] [7] and the use of spherical harmonics [6] [21]. More recently, these methods have been extended for anisotropic surfaces [35] [39].

While these advances have continued to improve ray-traced and radiosity images, hardware rendering quality has been relatively static with its basis on the Phong model. The hardware Phong model is not a true BRDF:

$$C_p = C_{me} + C_{sa}\,C_{ma} +$$

[2] Although we (and the hardware designers themselves) refer to the various implementations of exponent-based models as Phong's model, many are actually Blinn's[3] model. Usually mistakenly though of as equivalent, the two models are different albeit closely related[13]

$$\sum_l \frac{C_l}{K_{sdaf} + K_{sdav} D_{lp}} \left(C_{md} \left(\hat{N}_{pl} \cdot \hat{N}_p \right) + C_{ms} \left(\hat{N}_b \cdot \hat{N}_p \right)^{Emss} \right) \quad (2)$$

where the C values are the scene, material, and light colors, the $\hat{N}$ vectors represent the normal, bisector vector, and direction to light as seen in Figure 11, the K values are fixed and variable attenuation factors, and E_{mss} is the glossiness exponent. In fact, the Phong model does not guarantee conservation of energy as the specular term actually acts like a second diffuse term for low glossiness terms.

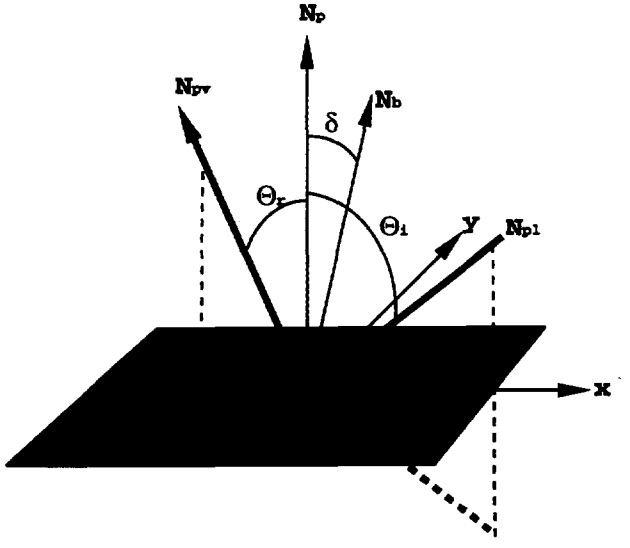


Figure 11: Phong BRDF Notation

While the traditional Phong representation is not founded in the physical transport of light, it can be made more accurate if fit to a physically-based model. Isotropic models provide a good reference scattering function close to the range of the Phong model. In particular we used the isotropic Gaussian model presented by Ward [36] as the basis to perform a Chi-Square fit of the Phong lighting model. This model provides a metric against real-world illumination in that it incorporates actual measured material parameters and has itself been tested against gathered data. The specular component of the Gaussian BRDF is

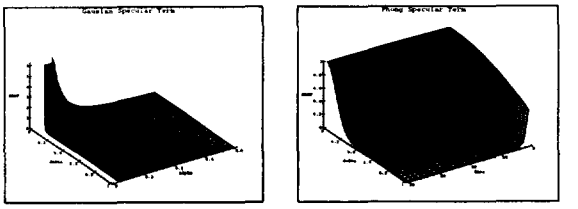
$$\rho_s \frac{e^{\left(-\frac{\tan(\delta)^2}{\alpha^2}\right)}}{\sqrt{|\cos(\theta_i)\cos(\theta_r)|}\,4\pi\,\alpha^2}, \quad (3)$$

and the corresponding Phong specular term rewritten from Equation 2 is

$$\rho_s \cos(\delta)^{Emss} \quad (4)$$

where δ is the polar angle between the half-vector and the surface normal as seen in Figure 11.

While the Phong model can be fit to the Gaussian BRDF, this fit model is not applicable to the hardware lighting model. If we examine the full diffuse and specular components of the hardware model, we note that $cos(\theta_i)$ is used



(a) Gaussian (b) Phong

Figure 12: Gaussian vs. Phong Specular Term

to modulate the light received by the angle incident to the surface normal by the source. The rendering equation (eq. 1) similarly uses a differential solid angle to represent the projected solid angle subtended by the source. This introduces an additional $sin(\theta_i)$ term not included in the hardware equation. While this term integrates out for a true point light source, it is present in Monte Carlo sampling systems such as Radiance [37] where there are only physically realizable sources. Therefore, to provide some consistency between the diffuse and specular computations as related to a physically-based system, the $sin(\theta_i)$ term is dropped from the rendering equation and the Phong model is fit to the rendering equation. The specular term therefore becomes

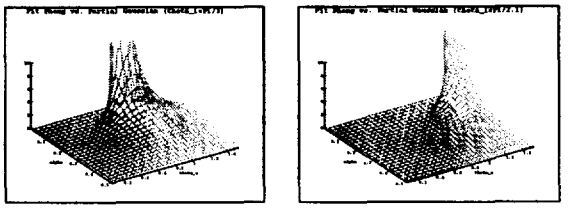
$$\cos(\theta_i)\,\rho_s \frac{e^{\left(-\frac{\tan(\delta)^2}{\alpha^2}\right)}}{\sqrt{|\cos(\theta_i)\cos(\theta_r)|}\,4\pi\,\alpha^2} \quad (5)$$

A Chi-Square two-parameter fit of this model produces the following Phong term:

$$.0398 \left(1.999 \frac{1}{\alpha^2}\right) \rho_s \cos(\delta)^{\left(1.999 \frac{1}{\alpha^2}\right)} \quad (6)$$

with a corresponding χ^2 value of 0.00008003.

The exactness of this fit is seen graphically for several θ_i in Figure 13.



(c) $\theta_i = \pi/3$ (d) $\theta_i = \pi/2.1$

Figure 13: Partial Gaussian vs. Fit Phong

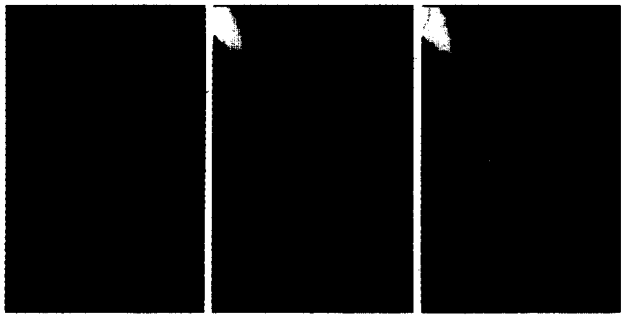
While this Phong model is founded in a more physical basis, it does not include the grazing effects of the original Gaussian. It tends to overstate specular highlights for acute θs, and conversely understates obtuse grazing angles, as detailed in [9]. Even with these limitations, the benefits of the fit model are further magnified by the ability to now incorporate measured material parameters as in [35], thereby providing physics-based material properties.

6 INDIRECT ILLUMINATION

Multi-pass Pipeline Rendering is best-suited for specular environments. As mentioned previously, radiosity is well-suited for calculation of global illumination effects for diffuse environments. It does not, however, capture the specular illumination which ray tracing and MPR are capable of. In addition, it does not readily support having dynamic environments[33]. The combination of MPR with a radiosity solution provides a solution for dynamic environments, both specular and diffuse.

While a radiosity solution does include the direct diffuse illumination of the environment, this is simply the first iteration or "bounce" of the total radiosity solution. If this first iteration is discarded from the final solution, the radiosity renderer produces the global indirect illumination of the environment. If no ambient term is included, this provides a linearly independent lighting calculation which can be simply added to the lighting calculations as defined above. This method is better suited for dynamic environments than other two-pass methods [33][27] in that the global direct component is not incorporated in the first pass (direct illumination) solution and the indirect component is not as affected by moving geometry.

The addition of the radiosity contribution readily occurs in image-space as demonstrated by Dorsey *et. al.*[12]. This method, however, can prove more complicated if specular surfaces are present. The radiosity image typically will not contain secondary images as in a mirrored image, and therefore the entire image cannot be added to the pipeline rendering image. This region needs to be first masked to prevent blending of one image containing the specular image (the pipeline rendering one) with one image without specular reflections (the radiosity one).



(a) Indirect (b) Direct (c) Full [(a)+(b)]

Figure 14: Calculating Indirect Illumination Image

A more flexible solution is to incorporate the indirect radiosity solution into the hardware rendering process instead of adding it to the final image. The indirect radiosity vertex coloring is rendered using the hardware shading in the first pass of the pipeline process. This replaces the ambient-only rendering stage. This obviates the specular image problem described above since specular images are generated also during the ambient-only stage which the indirect solution replaces. The direct solutions then use the original attribute information for all subsequent calculations, ignoring the radiosity vertex colorings. The hardware blending functions detailed previously perform the composition of the individual illumination effects, as seen in Fig. 14.

7 TOTAL SCENE ILLUMINATION

The previous sections detailed the individual components of the scene illumination model. It is the combination of these techniques which produce a final image. While we have briefly discussed the individual errors which result from hardware limitations, physical approximations, and performance tradeoffs, it is only through evaluation of the images produced by these methods that we can gauge the effectiveness of these techniques as a system.

In the previous sections, the individual components of a multi-pass rendering system were introduced. This includes shadow generation for real and virtual light sources and specular image generation from a virtual viewpoint. Both features rely on manipulating the hardware matrix stack to create these virtual positions. Both features require multiple passes to render for these virtual positions. Additionally, both features use stenciling to mask to appropriate screen regions for each rendering pass. This section examines the interaction of these two similar rendering processes.

7.1 Recursion

The coordination of the separate processes is seen in the following pseudo-code.

```
draw_window(Camera)
{
  if (SPEC_ON)            //if specular enabled
    draw_spec_objects(Camera); //  -draw specular view
  if (SHADOWS_ON){        //if shadows are enabled then..
    turn_lights_off();    // -turn all lights off
    draw_objects();       // -draw diffuse objects unlit
    for (each light)      // -loop over each light and..
      make_shadows();     //   -draw in light volume
  }else                   //else
    draw_objects();       //  -draw diffuse objects lit
}


draw_spec_objects(Camera)
{
  for (each spec_face){  //loop over all specular faces
    mask_face(spec_face);//-set stencil area for face
    reclassify_camera(Camera);//-move to new viewpoint
    draw_window(Camera); //-recursively draw new view
  }
}

make_shadows(light)
{
  env_sten(AFT_SPEC);     //objects inside light volume
  if (shadow_level>0)     //if shadowing a light volume
    env_sten(PRE_SPEC);   //objects outside light volume
  turn_light_on(light);
  draw_objects();         //add light (except shadows)
  turn_light_off(light);
  make_spec_shadows(shadow_level++);//add virtual light
}

make_spec_shadows(light)
{
  for (each specular_face){ //loop over specular faces
    spec_light(light,spec_face); //-move to new position
    make_light_volume(light);  //-mask light volume area
    make_shadows;           //  -create in light volume
  }
}
```

7.2 Light Accumulation

To provide correct accumulation, individual light contributions are added in RGB space at every iteration based on the linearity of light transport[11]. The first-pass rendering is performed with ambient light only; successive iterations are performed with no ambient light. By performing all illuminated renderings with a Z-buffer comparison function checking for equal values, only those visible object pixels are re-rendered and blended with the final image. While our practical experience with the system has not demonstrated many dynamic range problems due to the RGB-space limitations, the authors note that an extended color space representation such as presented by Ward[34] could greatly contribute to more physically-realistic images.

7.3 Stenciling

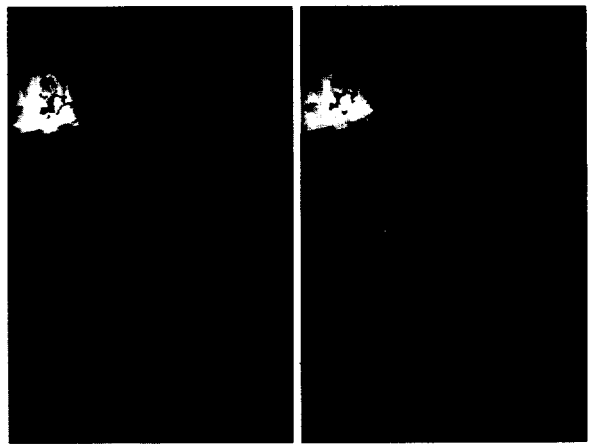
Both the rendering of shadows and the rendering of specular images require use of the stencil planes. For specular image generation, the stencil plane not only masks the valid rendering area, but also acts as a recursive counter to enable depth-first traversal in the recursive rendering process. At each level of recursion, shadows must be drawn in the valid area. This area may include previously rendered specular surfaces so that shadows may be cast on partially specular objects. These surfaces are rendered first as purely specular and then blended with their respective material properties. As described above, specular surfaces have their stencil values reset to zero after their specular image has been rendered. The valid rendering area is therefore popped back to zero, with lower recursive depths maintaining higher stencil values.

While it might seem that the zero specular stencil mask value would be a logical choice for the zero value in the plus-minus shadow algorithm, this is not the case. In order to have recursive reflections and refractions, we instead choose a value which is three-fourths of the maximum stencil value for our "zero" shadow value (SHAD_ZERO), and one-half of the maximum for our minimum shadow stencil value (SHAD_MIN). This provides half of the stencil buffer for shadow calculation and half for recursive specular levels. These values can be adjusted according to the recursion level needed or the shadow object complexity.

7.4 Comparisons With Ray Tracing

Fig. 15 demonstrates a comparison between an image produced through the Radiance[38] system and one produced through the pipeline rendering process. The pipeline image uses the described techniques with a constant ambient lighting of 0.10 replacing the inclusion of a radiosity solution indirect component. The first image shows the reference Radiance image which required 220 sec., and the second shows the pipeline image generated in 8.5 sec.

The RMS error between the two images is under 5% over the range of luminance values versus over 10% error for traditional single-pass rendering. Further accuracy can be gained using the radiosity-generated indirect component as described above, however the pre-processing computation cost often outweighs the increased realism benefit. The indirect component required 2224 patch shootings for a total of 2343 CPU seconds. This high computation cost, as with any radiosity-based system, prevents incorporation into a fully-dynamic interactive system. For a low ambient scene



(a) Radiance Image (b) Pipeline Image

Figure 15: Gaussian (Radiance) vs. Phong (MPR) Illumination

which is dominated by specular reflections, a constant ambient term should suffice.

8 PERFORMANCE

The preceding section detailed the coordination between the MPR techniques including shadow generation and specular surfaces. As both of these features proceed recursively, performance is primarily determined by the number of rendering passes (including both scene and shadow volume rendering); image size has minimal effect as compared to ray tracing. For Fig. 16 consisting of 40,463 polygons, the scene required 96 shadow generation passes and 49 scene rendering passes which together represent 92% of the total rendering time of 16.6 seconds for the 623x942 image. A 1270x995 image required 24.8 seconds. The corresponding radiance images without textures required 236 and 486 seconds, respectively. All renderings were performed on a single-processor 200MHz R4400 Onyx Reality Engine. Minimal shadow volume culling and poorly-modeled objects were used to demonstrate application on a non-"hand-crafted" environment. An environment of equal quality with "cleaner" models (i.e. interior, non-visible polygons removed) was rendered four times as fast.

A breakdown of the rendering process is seen in Table 1. As can be seen by this data, specular image generation and shadow generation can become very costly for highly specular environments for large depths of recursion. While much of this rendering overhead could be eliminated by more accurate viewpoint culling (i.e.[24][1]) for both specular image and shadow generation, some is inherent in the nature of the process. Yet even this cost can be reduced if some sacrifice of image quality or accuracy is permitted. The following sections discuss this quality/performance tradeoff.

8.1 Shadows Recursion

Shadows have been shown to be an important visual cue for producing realistic and understandable images. For an environment with n specular surfaces each visible to the others, shadows are highly dependent on the depth d of the recursive light reflections and refractions. As shadow generation

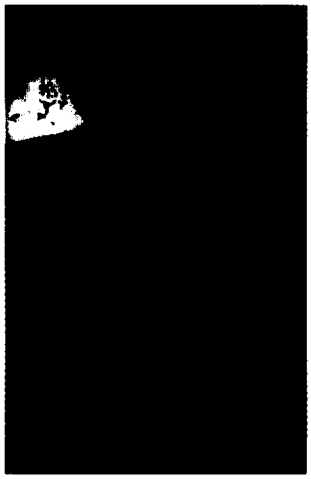


Figure 16: Pipeline Image

Pass	# Iterations	Time
Shadow Volume Rendering	109,662	6.89
Object Rendering	1,982,736	6.34
View-in-volume Check	6428	0.62
Misc. Checks	3836	0.26
Total (stored shadows)	N/A	16.61
Shadow Volume Generation	21,009	6.95
Total (calc shadows)	N/A	23.63

Table 1: Frame rendering statistics for bathroom environment.

is required for all virtual light sources resulting from the specular bounces as well as shadows which occur from the original light source but are then bent,

$$O(n\left\lceil\frac{d^2+3d}{2}\right\rceil+1) \tag{7}$$

shadow generations are required per image per light.

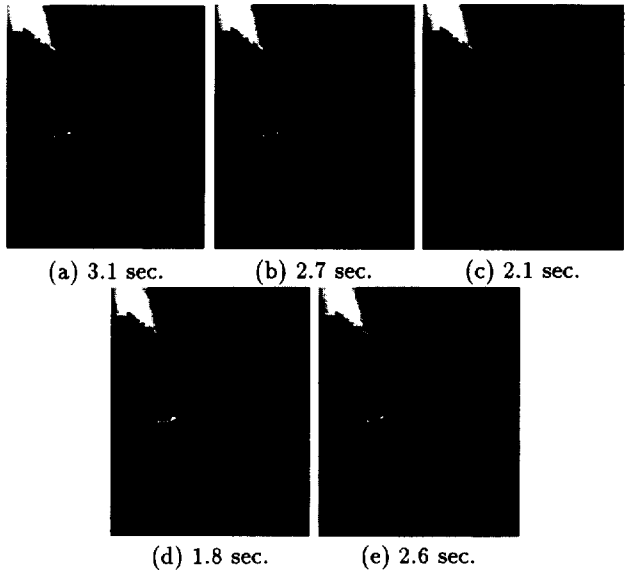
Often, however, shadows are desired to simply provide visual cues and some level of realism, and are not required to be completely accurate. In these situations, the number of rendering passes can be greatly reduced as can the memory overhead.

Fig. 17(a) shows the lighting passes for a single light source with one reflective and one refractive surface. In Fig. 17(b), shadow volumes which encounter more than one specular surface are eliminated. This effect is apparent in the shadow pattern on the tub. Fig. 17(c) shows the original situation without reflected and refracted shadows. Note the light volume is larger because the frame no longer blocks

Fig. 17	d	Spec Shad	2+ Bounces	# Passes* (scene/shad)	Total (sec)
(a)	2	YES	YES	17/32	3.1
(b)	2	YES	NO	17/28	2.7
(c)	2	NO	NO	17/16	2.1
(d)	1	N/A	N/A	9/12	1.8
(e)	1	YES	N/A	13/20	2.6

Table 2: Varying Shadow Settings

*Represents actual number reduced through visibility testing.



(a) 3.1 sec. (b) 2.7 sec. (c) 2.1 sec.

(d) 1.8 sec. (e) 2.6 sec.

Figure 17: Varying Shadow Settings

light from reaching the mirror, and the shower door handle no longer casts a shadow through the door. Fig. 17(d) replaces the refractive shower door with a purely transparent, non-refracting surface. Finally, Fig. 17(e) demonstrates the original scene with a shadow recursion depth of one instead of two. A summary of the required number of rendering passes is seen in Table 2.

8.2 Specular Image Recursion

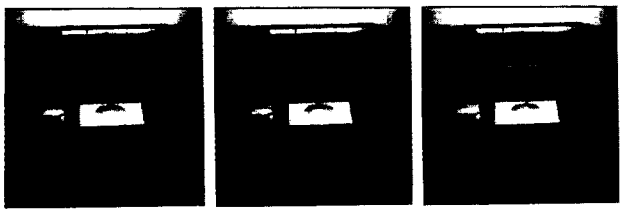
Like shadows, specular images also provide a measure of depth perception as well as added realism. In a complex specular environment, the interaction of specular surfaces can require many rendering iterations. Yet, like shadows, this interaction is often not required to have physical realism. Because shadows are generated within each specular image as well as recursively for each specular surface, the number of specular surfaces and recursion depth proves doubly important. For complete shadow rendering, shadow volume passes are required for shadows generated both before and after each specular surface encountered for the light source at that recursive level. For n specular faces at a recursive image depth of d,

$$O(n^d+1) \tag{8}$$

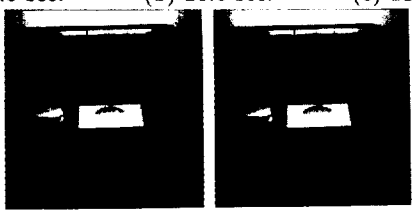
images are generated. Again, viewpoint culling can *greatly* reduce the actual number of passes required.

The images in Fig. 18 and corresponding Table 3 demonstrate the relationship between image quality and the performance for varying specular environments. Fig. 18(a) shows three specular surfaces (mirror, shower door, and floor) at a specular depth of two, and Fig. 18(b) a specular depth of one. In Fig. 18(c), the partial specular floor image has been excluded at a depth of two. This is then repeated at a depth of one in Fig. 18(d). In Fig. 18(e), the refractive shower door is replaced with a non-refractive transparent surface at a depth of two.

As can be seen, the rendering times range greatly yet many of the images are similar. In an interactive environment, complete rendering may not be needed until motion



(a) 63.5 sec. (b) 28.0 sec. (c) 11.4 sec.



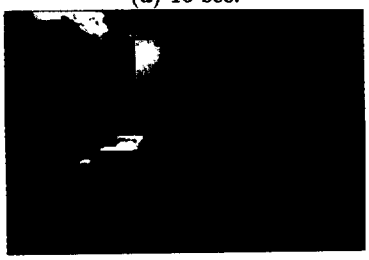
(d) 6.9 sec. (e) 3.4 sec.

Figure 18: Varying Specular Settings

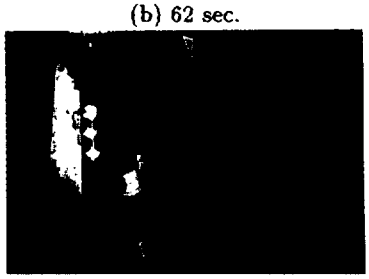
Fig. 18	n	d	Scene Iter #(sec)	Shad Iter #(sec)	Time (sec)	Rad Time (sec)	Perf
(a)	3	2	321(7.6)	760(53.5)	63.5	367	6x
(b)	3	1	129(3.1)	304(23.8)	28.0	236	8x
(c)	2	2	81(1.9)	160(9.2)	11.4	238	21x
(d)	2	1	49(1.1)	96(5.4)	6.9	211	30x
(e)	1	1	17(0.5)	24(2.8)	3.4	180	53x

Table 3: Varying Specular Settings (with associated Radiance times)

(a) 15 sec.



(b) 62 sec.



(c) 38 sec.

Figure 19: Varying Scene Geometry and Lighting

stops. This method is ideal for a progressive refinement situation, which could substitute from (e) to (c) to (a) as the necessary frame-rate drops (i.e. the camera slows). The switching criteria and ordering are an open issue.

8.3 Scene Dynamics

As mentioned previously, shadows maintain object associativity information. Therefore, unlike in a radiosity solution, moving scene geometry has little effect on rendering times. For the sequence of images in Fig. 19 from an animated sequence, the movement of the mirror required the most recomputation (6.1 sec/frame) as it affected virtual light source shadows. The movement of "Bob" required less recomputation (4.9 sec/frame). The entire animated sequence of 450 frames required less than 6 hours of rendering time.

9 CONCLUSION

With Multi-pass Pipeline Rendering (MPR), we have presented a platform for bridging the gap between static offline rendering systems and dynamic hardware-based graphics. We demonstrated a practical implementation of shadow and light volumes and incorporated this into a recursive paradigm permitting interaction with specular surfaces. This includes the specular direct light-volume component similar to Radiance's[38] light source reclassification for planar specular surfaces. We showed an extension to projected textures to approximate complex material transmission, as well as a method to render light-scattering materials themselves. We have tried to wrestle as much physical realism out of the lighting model itself without compromising performance. Consistent with the broad spectrum of achievable quality, we also presented a method to even include indirect lighting effects. While some of the components we use are based on existing techniques, we have provided a cohesive framework for extending and combining them with new rendering methodologies to produce ray trace-quality renderings. Even with minimal viewpoint culling, this pipeline rendering method demonstrated performance rates 5 to 50 times that of ray tracing for our test environments. Future work will focus on taking full advantage of available culling strategies for dynamic environments.

This paper intends not only to demonstrate the quality of effects achievable through pipeline rendering, but also to serve as a call for more focus on the use of the graphics hardware to perform realistic rendering. Indeed, many more extensions of this methodology are possible with current hardware, from achieving a better lighting model through individual vertex normal modulation to use of multiple processors. With other hardware platforms (i.e.[14]) and future hardware systems, parallel rendering pipelines may be able to exploit the independent nature of our multi-pass process. In this vein, this paper also calls for more open, accessible hardware pipelines which provide access in ways which the developers may never had imagined or intended.

References

[1] J. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations.* PhD thesis, UNC Chapel Hill, 1990.

[2] J. Arvo. Applications of irradiance tensors to the simulation of non-lambertian phenomena. In Robert Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 335–342, 1995.

[3] J. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):192–198, July 1977.

[4] J. Blinn and M. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19:542–546, 1976.

[5] L. Brotman and N. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Applications*, 4(10):71–81, October 1984.

[6] B. Cabral, N. Max, and R. Springmeyer. Bidirectional reflection functions from surface bump maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 273–281, July 1987.

[7] R. Cook and K. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[8] F. Crow. Shadow algorithms for computer graphics. volume 11, pages 242–248, July 1977.

[9] P. Diefenbach. *Pipeline Rendering: Interaction and Realism through Hardware-Based Multi-Pass Rendering*. PhD thesis, Dept. of CIS, U. of Penn., 1996.

[10] P. Diefenbach and N. Badler. Pipeline rendering: Interactive refractions, reflections, and shadows. *Displays (special issue on Interactive Computer Graphics)*, 15(3):173–180, 1994.

[11] J. Dorsey. *Computer Graphics Techniques for Opera Lighting Design and Simulation*. PhD thesis, Cornell University, 1993.

[12] J. Dorsey, J. Arvo, and D. Greenberg. Interactive design of complex time dependent lighting. *IEEE Computer Graphics and Applications*, 15(2):26–36, March 1995.

[13] F. Fisher and A. Woo. R.E versus N.H specular highlights. In Paul Heckbert, editor, *Graphics Gems IV*, pages 388–400. Academic Press, Boston, 1994.

[14] H. Fuchs, J. Goldfeather, J. Hultquist, S. Spach, J. Austin, F. Brooks, Jr., J. Eyles, and J. Poulton. Fast spheres, shadows, textures, transparencies, and image enhancements in Pixel-Planes. In *Computer Graphics*, volume 19, pages 111–120, July 1985.

[15] H. Fuchs and J. Poulton. Pixel-planes: a VLSI-oriented design for 3-D raster graphics. *Proc. of the 7th Canadian Man-Computer Communications Conf.*, pages 343–347, 1981.

[16] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):309–318, August 1990.

[17] X. He, K. Torrence, F. Sillion, and D. Greenberg. A comprehensive model for light reflection. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):175–186, July 1991.

[18] P. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, U. of California, Berkley, 1989.

[19] P. Heckbert and P. Hanrahan. Beam tracing polygonal objects. *Computer Graphics*, 18(3):119–127, 1984.

[20] J. Kajiya. The rendering equation. In *Computer Graphics*, volume 20, pages 143–150, August 1986.

[21] J. Kajiya and B. Von Herzen. Ray tracing volume densities. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 165–174, July 1984.

[22] D. Kay and D. Greenberg. Transparency for computer synthesized images. *Computer Graphics (SIGGRAPH '79 Proceedings)*, 13(2):158–164, August 1979.

[23] E. Kingsley, N. Schofield, and K. Case. Sammie. *Computer Graphics*, 15(3), 1981.

[24] D. Meagher. *The Octree Encoding Method for Efficient Solid Modeling*. PhD thesis, Electrical Engineering Dept., Rensselaer Polytechnic Institute, 1982.

[25] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. NBS Monograph 160, National Bureau of Standards, U.S. Dept. of Commerce, October 1977.

[26] B. Phong. Illumination for computer-generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[27] C. Puech, F. Sillion, and C. Vedel. Improving interaction with radiosity-based lighting simulation programs. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24, pages 51–57, March 1990.

[28] W. Reeves, D. Salesin, and R. Cook. Rendering antialiased shadows with depth maps. In *Computer Graphics*, volume 21, pages 283–291, July 1987.

[29] E. Schmid. *Beholding as in a Glass*. Herder and Herder, New York, 1969.

[30] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texutre mapping. *Computer Graphics*, 26(2):249–252, 1992.

[31] K. Torrance and E. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of Optical Society of America*, 57(9), 1967.

[32] D. Voorhies and J. Foran. Reflection vector shading hardware. In *Proceedings of SIGGRAPH '94*, pages 163–166, July 1994.

[33] J. Wallace, M. Cohen, and D. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *Computer Graphics*, volume 21, pages 311–320, July 1987.

[34] G. Ward. Real pixels. In J. Arvo, editor, *Graphics Gems II*, pages 80–83. Academic Press, San Diego, CA, 1991.

[35] G. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):265–272, July 1992.

[36] G. Ward. Measuring and modeling anisotropic reflection. In *Computer Graphics*, volume 26, pages 265–272, July 1992.

[37] G. Ward. The radiance lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):459–472, July 1994.

[38] G. Ward. The RADIANCE lighting simulation and rendering system. In *Proceedings of SIGGRAPH '94*, pages 459–472. ACM Press, July 1994.

[39] S. Westin, J. Arvo, and K. Torrance. Predicting reflectance functions for complex surfaces. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):255–264, July 1992.

[40] L. Williams. Casting curved shadows on curved surfaces. In *Computer Graphics*, volume 12, pages 270–274, August 1978.