PHATextures: A Surface-Based Data Structure for Interactive Rendering of Glossy Surfaces

Justin Hensley Josh Steinhurst Greg Coombe Anselmo Lastra

Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, USA

{hensley, jsteinhu, coombe, lastra}@cs.unc.edu

Abstract

A good use for the raw computation power of the next generation of graphics hardware is global illumination. Historically, global illumination algorithms have been executed on general purpose CPUs that are able to support complex data structures easily. The performance advantage of graphics hardware lies in its regular and independent computational structure. To retain this advantage there is a growing need for hardware friendly data structures that support non-traditional uses of the GPU such as global illumination. We propose a hardware friendly data structure for the interactive rendering of glossy surfaces by storing a pre-computed, view-independent representation of the incident radiance. The data structure and rendering algorithm approximates the Monte Carlo integration techniques commonly used in global illumination algorithms. The algorithm is applied to photon mapping and achieves interactive results on current graphics hardware. Due to its nature, the data structure is well suited to dynamic updates.

Keywords: Graphics Hardware; Global Illumination.

1 Introduction

Glossy surfaces are all around us. They are difficult to render interactively because, unlike diffuse surfaces, their reflection of the world is view dependent. To generate realistic images, the renderer must take into consideration light striking the surface from all directions.

The reflected radiance is determined by convolving all the light arriving from the entire hemisphere of directions with the BRDF. Many global illumination algorithms, such as Monte Carlo ray tracing and photon mapping, use a recursive Monte Carlo integration to estimate the incident radiance [Kajiya 1986, Jensen 2001]. Non-trivial scenes require a large number, often hundreds, of directions to be sampled to have an acceptably low error in the estimate of incident radiance. Unfortunately each sample can be very costly to calculate.

We are investigating data structures for use in future global illumination hardware, in the same spirit as [Ma 2002]. Instead of a spatial (3D) data structure, such as the k-d tree used by photon maps, we present a surface-based data structure. We believe that this approach is better suited for current and future hardware implementation.

PHATextures (Photon Accumulation Textures) are a sparse sampling of the light arriving across a surface from all directions. They store a per-texel, pre-computed incident radiance sample. Each sample consists of energy and an incident direction. During rendering, we combine spatially close samples to approximate a Monte Carlo integration.

Unlike other techniques for interactively rendering of glossy surfaces under indirect lighting, PHATextures lend themselves to easy updates. Each PHATexel can be replaced independently. Dynamic scenes can be handled by ensuring that all PHATexels are updated within some amount of time similarly to frameless rendering [Bishop 1994]. This allows the light transport simulation to be decoupled from rendering.

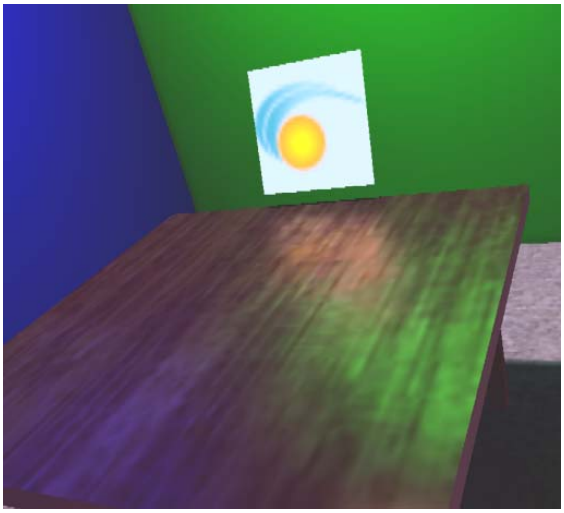

Figure 1: A glossy tabletop reflects a textured object as well as the walls. This scene runs at interactive rates on current graphics hardware.

Indirect illumination, in the absence of strong caustics, will have low spatial variation. The quality of the results can be improved for most scenes by separating out the contribution of the direct light and storing only indirect illumination in the PHATextures.

After reviewing related work and providing background on Monte Carlo integration, we describe PHATexture generation and rendering. We then show how to create PHATextures from a photon map [Jensen 2001]. A PHATexture rendering algorithm running on current graphics hardware is described, followed by our results and conclusion.

2 Related Work

Surfaces are often characterized as diffuse, glossy or specular. We first review ways to render specular and diffuse surfaces, and then concentrate on the subject of this paper: glossy surfaces. Finally we discuss representations for storing radiance in a model.

Specular surfaces can be efficiently rendered interactively with dynamically updated environment maps, at the cost of some geometric inaccuracies. An alternative is ray tracing, which can be computed at interactive frame rates using parallelism and other optimizations [Parker 1999; Wald 2001]. It is even possible to ray trace using graphics hardware [Carr 2000; Purcell 2002].

Diffuse surfaces can be rendered from a pre-computed lighting solution by either storing the results as vertex colors, or on textures [Cohen 1993]. With current hardware it is possible to compute a radiosity solution at near-interactive rates [Coombe 2003].

A technique for rendering glossy surfaces is the surface light field [Wood 2000; Chen 2002]. This method reduces a very dense set of exitant radiance information to a small set of parameters that are defined over a surface. For each point on the surface the parameters are interpolated and define a two-dimensional function that can be evaluated for a given viewing direction to calculate the reflected radiance. The results are very good, but require a very dense set of input data and an expensive non-linear fit that must be applied to the entire surface as a whole. Computation of the input is significant, as they are usually constructed from captured images, making construction from full global illumination algorithms difficult. Dynamic update is complicated by the non-linear fit over many texels.

Similar work is the eigen-texture method of [Nishino 1999]. Other parametric methods that are not image-based and can render glossy BRDFs, with some limits on the material properties, include those of [Heidrich 1999; Kautz 1999; Kautz 2000; McAllister 2002]. These methods have been employed on graphics hardware with a fixed number of lights and direct lighting. View-dependent texturing [Debevec 1996; Heigl 1999; Buehler 2001] is another possible approach, although may need many texture samples to capture the directional nature of glossy materials. This technique sees the most use in image-based rendering of real-world scenes.

Several caching techniques have been presented [Larson, 1998; Larson, 1999; Walter 1999; Stamminger 2000; Simmons 2000; Dmitriev 2002]. These techniques are effective, and produce some of the best results for interactive display of global illumination solutions. However, it is not clear that such complicated methods are a good way to approach a future hardware solution.

Surface-based data structures to store directional representations of energy include those of [Immel 1986] who discretized the hemisphere to represent the directional contributions of light, [Sillion 1991] who used spherical harmonics, and [Christensen 1996] who used wavelets. An alternative approach that makes search in a 3D data structure potentially hardware-friendly is the block hashing scheme of [Ma 2002]. This in contrast to the surface-based approach that we present in this paper.

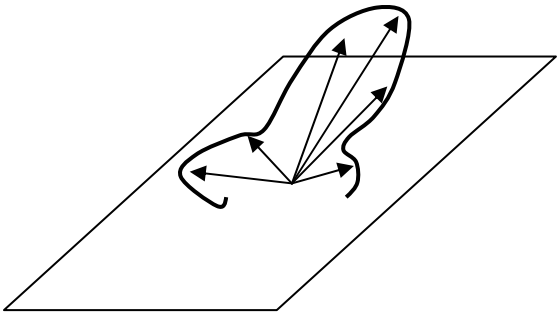

Figure 2: Monte Carlo integration of a hemisphere at a point. The function is sampled in many directions. Convergence is improved by concentrating the samples in areas of high value.

3 Background

3.1 Monte Carlo Integration

Monte Carlo integration is a powerful set of techniques for integrating functions, especially ones with sharp discontinuities. The function is numerically evaluated at random points over the domain of interest. These samples are averaged and will, in the limit, approach the value of the true integral. We are interested in integrating the light arriving at a surface point over the entire hemispherical domain. Figure 2 shows a set of samples and how they are used to estimate the incoming light.

The rate of convergence for Monte Carlo methods with truly random sampling of the function is inversely proportional to the square root of the number of samples. A very large number of samples may be required in order to reduce the variance to acceptable levels so that the estimate will converge with a reasonable amount of samples. The two most popular strategies for lowering the variance of the samples are stratification and importance sampling. These techniques try to intelligently focus the sampling. Stratification will insure that the entire domain is examined at a minimum frequency. This will prevent a large spike in the function from dominating the final result.

A more powerful strategy for improving the convergence of Monte Carlo integration is importance sampling. Instead of a random or stratified random sampling of the hemisphere, the function is sampled in those directions that will have the most significant impact on the final result. With prior knowledge of the function being integrated in Figure 2 it is possible to sample the strong lobe more closely. The low power portions of the function contribute little to the integral so more error can be tolerated. Two things define how important a sample at a point x will be: 1) how greatly the local BRDF at x will weight samples from a particular direction ω; and 2) the magnitude of the light arriving from that direction. For a glossy surface, the BRDF is unable to provide much guidance unless we know where the camera is. This makes pre-processing of glossy surfaces difficult.

4 PHATextures

A PHATexture is a sparse sampling of the light arriving at a surface from all directions. Each PHATexel on surface i at point x, $P_i(x)=\{L(x, \omega), \omega)$, is an estimate of the radiance L received at point x from the single

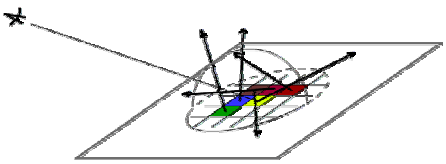

Figure 3: A neighborhood of PHATexels provides an approximation of the incident radiance at any point on a PHATexture suitable for use in Monte Carlo integration.

direction ω. There are five components to a PHATexel, the incoming angle (θ, φ) and three color channels (RGB). Figure 3 shows a diagram of PHATexels on a surface. The left half of Figure 4 shows the RGB components of the PHATexture used to render that image.

The key to the efficient use of PHATextures is the selection of the sampling directions at the texels. They must sample the entire incoming radiance function well in order to provide the basis for a good Monte Carlo estimate.

4.1 Generating PHATextures

PHATextures are initialized during a pre-process and can be incrementally updated. This enables the system to use a relatively expensive function to compute the sample of estimated incident radiance. The locations of the PHATexels on the surface being sampled are fixed by the scene geometry and the PHATexture resolution. For each PHATexel a single sampling direction is chosen and the incident radiance from that direction is calculated using a global illumination algorithm. The choice of algorithm is independent of our method; in the next section we present a system using photon mapping.

PHATexels that are spatially close must sample radiance from different directions so that the neighborhood around each PHATexel will contain a diverse set of directions. This constraint is equivalent to ensuring that the samples used in a Monte Carlo integration are representative of the domain of interest. Stratification and importance sampling are both effective ways of handling this. Importance sampling is expected to give higher quality results for the same number of samples.

We have found that it is better to store only indirect illumination in a PHATexture and add the direct lighting back in. There are three reasons for this: 1) direct illumination usually has significantly more spatial variation than indirect lighting; 2) there are many more computationally efficient methods for calculating it; and 3) by removing the direct illumination from consideration we are able to store more indirect information per PHATexture.

4.2 Evaluating PHATextures

To render the scene, we need to estimate the radiance reflected towards the eye at each visible point, x. The reflected radiance is a convolution of the incident radiance with the local BRDF. In some scenes this is a very simple process. When there is only direct illumination from a single point source for example, just one sample of irradiance is needed and it can be easily computed for each surface point from the power and position of the light source. A single sample of irradiance is sufficient because that is the only light that can possibly arrive on the surface. However, in the presence of indirect light a single estimate of irradiance[1] will be insufficient to accurately calculate the reflected radiance for all but trivial scenes. Jensen [Jensen 2001] found that a large number of directions are needed to calculate the reflected radiance with low error, which is essential for high quality images.

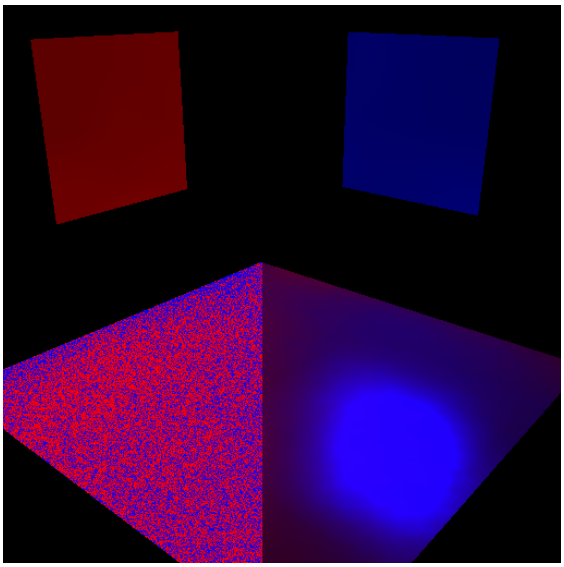

Figure 4: Each PHATexel stores the incident radiance in a specific direction; the RGB values are shown. A diversity of directions in all neighborhoods of this surface allows for the interactive rendering of glossy reflections on the right.

By collecting data from the neighborhood of PHATexels around x (see Figure 3) a large number of diverse radiance samples are obtained. When these samples have had the BRDF applied and are averaged together they form an estimate of the exitant radiance at x. The BRDF may contain texture information or be otherwise spatially varying. If direct light was excluded during the processing it can be added back now using any of the techniques available.

The radiance samples computed close to x are more likely to be reasonable then those that are further away. The contributions of each sample are weighted by distance. The size of the neighborhood gathered must be large enough so that it contains samples from a diverse

[1] Each PHATexel is specified to contain a single estimate of incident radiance. The standard formulation of the BRDF requires an estimate of irradiance arriving along a single angular direction. The sample of the radiance field may be converted into irradiance using the formula $E_\omega = 2\pi L(n \bullet \omega)$ where n is the surface normal. This conversion could be computed during preprocessing and stored in the PHATexture instead of the incident radiance, or alternatively the BRDF can be modified to take it into account.

set of directions, but small enough to prevent excessive blurring. Since too small a neighborhood was used in Figure 5, it exhibits a large amount of noise. If a compromise cannot be reached between noise and blur, then a higher resolution PHATexture is required in that area.

Gathering a neighborhood of PHATexels presents a problem when you reach the edge of the PHATexture. To ensure a stable support of the filter, the PHATexture is extended past the edge of the geometry. The extra PHATexels are used when the filter would otherwise be unsupported.

5 Photon Mapping Example

Photon mapping, described in [Jensen 2001], is a powerful technique for computing global illumination solutions of complex scenes. However, it is very costly to generate each image even with the addition of such techniques as irradiance caching [Ward 1992]. The computationally expensive operation is the recursive Monte Carlo integration of irradiance. We use PHATextures to accelerate this operation, and enable the interactive rendering of photon maps containing glossy surfaces. A ray tracing front end would be required for highly specular objects. After a quick overview of the fundamentals of photon mapping we describe the procedure for processing the computed photon map into PHATextures.

The photon map is generated by shooting photons, carrying flux, from light sources into the scene. The photons are probabilistically absorbed by partial or totally diffuse surfaces as they are traced through multiple surface interactions. The photons are stored in a 3D data structure for lookup during the rendering phase.

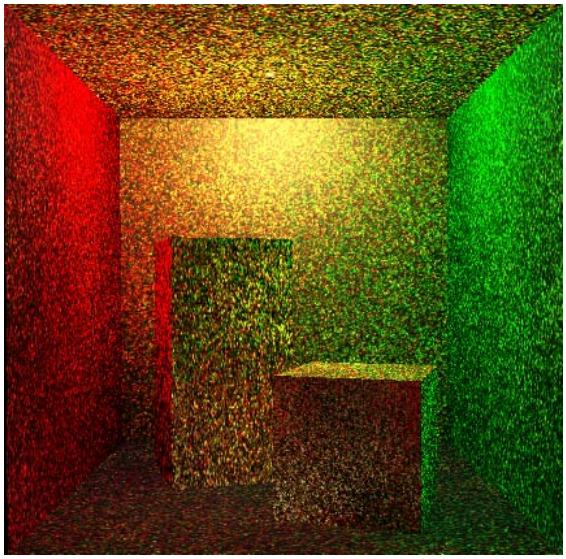

Figure 5: A neighborhood of PHATexels provides an approximation of the incident radiance at any point on a PHATexture. This image was rendered using a small neighborhood, effectively using too few samples in the Monte Carlo integration, which results in noise.

For each desired pixel during rendering, a ray is cast from the camera onto the scene, and the reflected radiance at that point is calculated. A scene with highly specular objects should utilize ray tracing since the reflected radiance function on a highly specular surface has very high frequencies. There are two basic algorithms for determining the reflected radiance at a surface point from a photon map.

The *approximate solution* is to gather a local neighborhood of photons on surface. The flux carried by the photons within a gather radius is summed, with appropriate weighting, to determine the irradiance. This irradiance is combined with the BRDF to determine the reflected radiance. This is a direct application of the Monte Carlo integration.

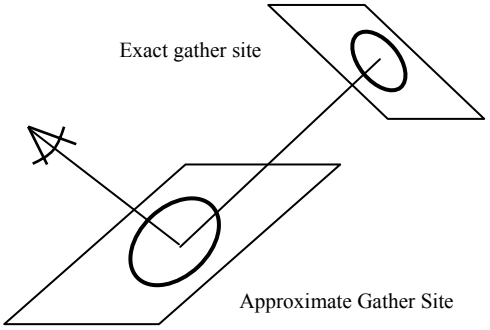

Figure 6: The two photon map rendering solutions differ by where photons are gathered to compute reflected radiance, the first bounce or the second.

A more correct answer, referred to by Jensen as the *exact solution*, uses Monte Carlo recursively. Many individual rays $\{x, \omega\}$ for every surface position x are traced in to the scene. At the first intersection along each ray the k-nearest photons are gathered and combined with the BRDF to compute the reflected radiance back along ω towards x. This is the same operation that is performed at the original surface point by the approximate solution. The Monte Carlo integration at x proceeds as before but using these higher quality samples as estimates of incident radiance. Computing each sample is now a costly operation that must be performed for many directions at x. Typically hundreds of rays per point x are required to generate low-noise images.

5.1 Creating PHATextures from a Photon Map

During the pre-process stage a photon map is used to construct a PHATexture for each glossy surface. Each photon has a position on a surface, a flux, and an incident direction.

5.1.1 The approximate solution analogue

The analogue to the approximate solution is to store photons directly in a PHATexture. If each texel represented a photon, then collecting a neighborhood of PHATexels would be equivalent to gathering photons. Unfortunately photons are deposited on surfaces neither uniformly nor directly at PHATexel locations, making it difficult to choose exactly one photon per PHATexel.

We have to make a choice between all of the photons that are deposited near to each PHATexel. We must not

consider photons that are arbitrarily far away from *x* or the spatial coherence assumption will no longer hold. Using a similar maximum gather radius to that used by the photon map gathering routine, we can establish a list of candidate photons for each PHATexel. If the list is empty then there is no irradiance stored at the PHATexel. If not empty, then only one photon can be chosen from the list of candidates.

The naïve strategy is to choose the nearest photon. However this will fail to generate PHATexels with sufficient diversity of angles in a given neighborhood unless there is a very high photon density. If photons are sparse around *x* then always choosing the nearest photon for each PHATexel will result in a very poor sampling of the hemisphere in the neighborhood of *x*. In fact the directions would all be identical. Increasing the number of photons traced through the scene is expensive because it is hard to fairly concentrate photons to land near a specific location.

A better solution is to consider all the photons within a gather radius and choose one of them probabilistically. We compute a given photon's weight at *x* using a cone filter. Photons that land exactly at *x* receive a weight of one, and contribution falls off linearly to zero. The probability that a photon will be chosen by a PHATexel is that photon's weight divided by the sum of the weights of all photons. This strategy intermingles the photons in areas of low density and will gradually reduce the influence of distant photons in areas with high density. Therefore sufficient diversity in angular sampling will be achieved.

Rather than explicitly constructing a list of candidate photons for each PHATexel, we splat [Westover 1990] each photon one at a time into the PHATexture. Only a single photon at a time may be stored in a PHATexel. Each photon has a chance to force out the currently stored photon. This chance is proportional to its weight divided by the sum of the weights of all photons that have already been splatted to this PHATexel. This method is invariant to the order of the photons. The last photon splatted has the same chance of being chosen as the first photon, assuming equal weights. The winning photon's flux is converted to incident radiance and stored along with the direction from which the photon came as the value of $P_i(x)$.

Unfortunately, the method just described does not generate satisfactory images due to the poor quality of the incident radiance estimate along the splatted direction. Also, it is unable to correctly handle areas of high photon density. If the number of photons in a portion of the surface exceeds the number of PHATexels available then some are left out, and the flux that they represent is lost.

Although this is not a good way to generate final images, we find that it is useful for importance sampling the directions. The photons that are chosen by the splatting technique are by definition likely to originate at the surfaces that reflect light onto the surface of interest. Objects that reflect a great deal of light to a surface will have a greater number of photons that point towards it.

5.1.2 The exact solution analogue

The estimates of incident radiance stored in the PHATextures by the method described in the previous section were easy to compute but were of low quality. The photon map exact solution produces very good results, but requires a very expensive ray casting and photon gathering for every direction at each point evaluated. The more work that is required to calculate the estimated value of incident radiance for each direction, the more striking the advantages of the PHATextures become.

To improve the convergence rate of the Monte Carlo integration we use importance sampling. The method of the previous section gives us information about the distribution of light arriving at x. Instead of directly using the winning photon's flux, we cast a ray in the direction from which the photon arrived. At the intersection of this ray with the scene we perform a photon gather, just like the exact photon map solution. The resulting reflected radiance is converted into incident irradiance and stored along with ω at $P_i(x)$. We use this method exclusively.

5.2 Implementation details

For many of the stages of our pre-process we use a modified open source photon mapping program [Jarosz]. Photons are traced out into the scene, reflecting and being absorbed by surfaces in the model. The position, energy, direction, and object ID of each photon is recorded in a list as they are deposited into the photon map. These collected photons are then splatted into the PHATextures to determine the importance-sampled direction. For each PHATexel a single ray is cast. At the intersection, a k-nearest neighbor search is performed and the gathered photons are used to calculate the radiance received at the PHATexel. During the splatting, but not the gathering, first-hit photons are excluded if direct illumination will be computed separately during rendering (using any of the standard methods).

6 Hardware Implementation

There are five components to each PHATexel, the incoming angle (θ, φ) and three color channels (RGB). The PHATextures are currently implemented as a floating-point texture with four channels. The two angles are packed into half-precision (16-bit) floating point values. Before storage we convert the incident radiance into irradiance.

Floating-point storage is necessary even when direct illumination is removed because there is a high dynamic range in the value of radiance. It is critical to maintain this range due to the glossy BRDF. Objects can have bright contributions from one direction and very dim contributions from another. The smaller value cannot be discarded on a glossy surface because for some viewing angles they may be more important then the high valued samples.

Once per frame, after the camera has been positioned, we apply the BRDF to each PHATexel to create an intermediate texture. This texture will be used to construct a mipmap as described below. The BRDF only has to be computed once per PHATexel per frame.

6.1 A mipmap cone filter

The naïve rendering algorithm described before collects the results from all texels within a gather radius around point *x* in the intermediate texture and sums them using a cone filter. As mentioned, many directional samples are required to acquire a good estimate, even with

importance sampling. Although the intensive global illumination calculations have already been performed, reading several dozen values out of texture memory applying the BRDF for every displayed pixel is prohibitively expensive on current hardware. Since our goal is real-time display, we can approximate these texture reads using mipmaps.

We construct a floating point mipmap from the intermediate texture. For each displayed pixel, bilinear samples are taken from several different levels of the mipmap. Each texel of the intermediate texture contributes to multiple levels of the mipmap. As Figure 7 shows, sampling different levels of the mipmap at the point of interest approximates a cone filter. This filter does add a few artifacts, however. The most noticeable are the crosshairs that arise from any use of a linear filter. Bi-cubic texture sampling would alleviate this, although at the expense of extra memory bandwidth that could be better spent on higher resolution PHATextures. The most serious error though is that the filter has a variety of square supports that are not centered at the pixel being shaded.

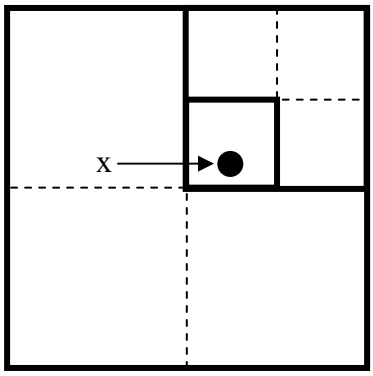

Figure 7: A neighborhood of PHATexels provides an approximation of the incident radiance at any point on a PHATexture.

6.2 Direct illumination and textures

If the scene only consists of point light sources and objects that cast no shadows, then the standard direct lighting can be easily implemented in graphics hardware. It is important to be sure that each light's energy is calculated correctly in the proper units so that the reflected radiance is compatible with the contents of the mipmap filter. Scenes with area lights or shadows can be computed using ray casting and pre-computed shadow maps. Once the total illumination is computed, standard shading algorithms can be used.

7 Results & Performance

The renderer was implemented on an Nvidia GeForce FX using three fragment programs. The first is used to create the intermediate texture, the second generates the floating-point mipmap levels and the third interpolates the floating-point mipmap, computes direct illumination from a point source and applies texturing. The first two programs must run on every polygon that has a PHATexture at the full resolution of the PHATexture. The final program, which involves many floating-point texture reads, is executed in proportion to the number of projected pixels.

The images presented through this paper were all generated at interactive rates after a view-independent pre-process. For the scenes in this paper the entire pre-process took under an hour.

8 Conclusions and Future Work

PHATextures allow for the interactive display of glossy surfaces with indirect illumination that has been previously calculated by a global illumination algorithm. The structure of the PHATextures allows for each PHATexel to be computed independently and updated as objects move or the lighting solution is enhanced

PHATextures are amenable to current graphics hardware, although they would benefit from greater texture bandwidth and larger memory sizes. As a surface based rather then a three-dimensional data structure the memory accesses are simple, conforming to traditional texture map address patterns, which would allow for relatively simple prediction and caching schemes.

The next step in this work is to implement a system that updates the PHATextures from a global illumination solution as objects move or the solution improves. The global illumination calculation can be decoupled from rendering. We plan to divide the work between the CPU and the GPU. The CPU will process the scene updates and perform the random access memory reads necessary for gather photons out of the photon maps. When results are calculated small updates will be rendered into the PHATextures that remain resident on the graphics hardware. The graphics system will be able to continuously render while the CPU is busy. An interesting alternative would be to do both calculation and rendering on the graphics hardware as [Coombe 2003] does.

We also plan to investigate architectures that compute the global illumination solution on the graphics hardware. We believe that, initially, the hardware will be too slow to compute a full solution at high frame rates. The PHATextures will act as a buffer to decouple the global-illumination update from the rendering.

9 Acknowledgements

The authors would like to thank the UNC Global Illumination reading group for their assistance and many discussions. This work was supported in part by NSF grant ACI-0205425 and a NSF Graduate Research Fellowship. Additional equipment was provided by NVIDIA.

References

Bishop, G., H. Fuchs, L. McMillan and E. J. S. Zagier (1994). Frameless Rendering: Double Buffering Considered Harmful. Proceedings of SIGGRAPH 94.

Buehler, C., M. Bosse, L. McMillan, S. J. Gortler and M. F. Cohen (2001). Unstructured Lumigraph Rendering. Proceedings of SIGGRAPH 2001, ACM.

Carr, N. A., J. D. Hall and J. C. Hart (2002). The Ray Engine. Graphics Hardware 2002, Saarbrucken, Germany.

Chen, W.-C., J.-Y. Bouguet, M. Chu and R. Grzeszczuk (2002). "Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields." ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002) 21(3): 447--456.

Christensen, P. H., E. J. Stollnitz, D. H. Salesin and T. D. DeRose (1996). "Global Illumination of Glossy Environments Using Wavelets and Importance." ACM Transactions on Graphics 15(1): 37-71.

Cohen, M. F. and J. R. Wallace (1993). Radiosity and Realistic Image Synthesis. Cambridge, MA, Academic Press.

Coombe, G., M. Harris, and A. Lastra (2003) Radiosity on Graphics Hardware. Submitted to Graphics Hardware 2003.

Debevec, P. E., C. J. Taylor and J. Malik (1996). Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. SIGGRAPH 96 Conference Proceedings. H. Rushmeier, ACM SIGGRAPH, Addison Wesley: 11--20.

Dmitriev, K., S. Brabec, K. Myszkowski and H.-P. Seidel (2002). Interactive Global Illumination using Selective Photon Tracing. 13th Eurographics Workshop on Rendering.

Immel, D., M. Cohen and D. Greenberg (1986). A Radiosity Method for Non-Diffuse Environments. Proceedings of SIGGRAPH 86.

Heidrich, W. and H.-P. Seidel (1999). Realistic, Hardware-Accelerated Shading and Lighting. Proceedings of SIGGRAPH 99.

Heigl, B., R.Koch, M. Pollefeys, J. Denzler and L. V. Gool (1999). Plenoptic modeling and rendering from image sequences taken by hand-held camera. Proc. DAGM 99.

Jarosz, W. http://www.renderedrealities.net/

Jensen, H. W. (2001). Realistic Image Synthesis using Photon Mapping, AK Peters.

Kajiya, J. T. (1986). The Rendering Equation. Proceedings of SIGGRAPH 86.

Kautz, J. and M. D. McCool (1999). Interactive Rendering with Arbitrary BRDFs using Separable Approximations. Eurographics RenderingWorkshop 1999.

Kautz, J. and H.-P. Seidel (2000). Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. SIGGRAPH / Eurographics Workshop on Graphics Hardware.

Larson, G. W. (1998). The holodeck: A parallel ray-caching rendering system. Second Eurographics Workshop on Parallel Graphics and Visualisation, Rennes, France.

Larson, G. W. and M. Simmons (1999). "The Holodeck Ray Cache: An Interactive Rendering System for Global Illumination in Nondiffuse Environments." ACM Transactions on Graphics 18(4): 361-98.

Ma, V. C. H. and M. D. McCool (2002). Low Latency Photon Mapping Using Block Hashing. Graphics Hardware 2002.

McAllister, D. K., A. A. Lastra and W. Heidrich (2002). Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. Graphics Hardware 2002, Saarbruecken, Germany.

Nishino, K., Y. Sato and K. Ikeuchi (1999). Eigen-Texture Method: Appearance Compression Based on 3D Model. Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99).

Parker, S., P. Shirley, Y. Livnat, C. Hansen and P.-P. Sloan (1998). Interactive Ray Tracing for Isosurface Rendering. IEEE Visualization 98.

Purcell, T. J., I. Buck, W. R. Mark and P. Hanrahan (2002). "Ray Tracing on Programmable Graphics Hardware." ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002) 21(3): 703-712.

Sillion, F. X., J. R. Arvo, S. H. Westin and D. P. Greenberg (1991). A Global Illumination Solution for General Reflectance Distributions. Proceedings of SIGGRAPH 91.

M.Simmons and C. H. Sequin (2000). Tapestry: A Dynamic Mesh-Based Display Representation for Interactive Rendering. Rendering Techniques 2000: 11th Eurographics Workshop on Rendering.

Stamminger, M., J. Haber, H. Schirmacher and H.-P. Seidel (2000). Walkthroughs with corrective texturing. Rendering Techniques 2000: 11th Eurographics Workshop on Rendering.

Wald, I., C. Benthin, M. Wagner and P. Slusallek (2001). Interactive Rendering with Coherent Ray-Tracing. Proceedings of EUROGRAPHICS 2001.

Walter, B., G. Drettakis and S. Parker (1999). Interactive Rendering using the Render Cache. Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering).

Ward, G. J. and P. Heckbert (1992). Irradiance Gradients. Third Eurographics Workshop on Rendering.

Westover, L. (1990). Footprint evaluation for volume rendering. Proceedings of SIGGRAPH 90.

Wood, D., D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. Salesin and W. Stuetzle (2000). Surface Light Fields for 3D Photography. Proceedings of SIGGRAPH 2000.