Rendering of Caustics with Photon Maps

Eric Li*
Andrew Dolce†
Rensselaer Polytechnic Institute

Abstract

A number of techniques exist for approximating the rendering equation in order to create realistic detailed images. Photon mapping is a method that allows for the accurate rendering of a wide range of reflective and refractive phenomena. We explore the basis of the photon mapping technique with the goal of constructing a system for rendering such effects, particularly in regard to caustics. Our implementation is divided into two separate passes, each of which has achieved a different degree of success.

1 Motivaion

Our project is motivated primarily by an interest in various methods of global illumination and in the stunning images that they create. In studying computer graphics, we have explored fairly simple rendering schemes involving ray tracing and radiosity, and yet none of our prior implementations included the proper handling of a variety of refractive effects, particularly in regard to caustics. Despite the fact that a large amount of research already exists for a variety of global illumination techniques capable of rendering refractive elements, we aim to successfully implement a system for rendering caustics using an illumination scheme reasonable for the scope of a student project.

2 Related Work

A wide range of research has been done in relation to global illumination techniques that support refractive phenomena. One method, referred to as eikonal rendering, involves approximations of the eikonal equation, which comprises the fundamental principle of geometric optics. In their 2007 paper, Ihrke et al. provide a generalized method for producing a variety of visual effects by using the eikonal equation to compute radiance along arbitrarily curved rays. Using a method of adaptive wavefront tracing, light is simulated by emitting spherical or planar particle-based wavefronts into the scene and storing radiance contributions in a grid of voxels. Although this method holds the advantage of being able to compute radiance locally for surface points in parallel, thus lending itself to an efficient implementation on the GPU, we felt that an eikonal illumination scheme was beyond a reasonable scope for our project.

A second global illumination scheme capable of rendering caustics and other refractive effects involves a method known as photon tracing. A common approach for this method, as proposed by Jenson in 1996, is called photon mapping and involves a two-step process. During the first step, photons are emitted from each light source into the scene in a manner similar to ray-tracing. When each photon intersects an object in the scene, the collision point is stored in a photon map, which Jensen implemented using a kd-tree. The second step involves gathering the radiance and rendering the scene. This is done via reverse ray tracing, computing the radiance at each encountered surface point by searching the photon map for appropriate neighbors and computing the summed radiance. In Jenson's implementation, the emission step involved first casting uniformly distributed photons into the scene and storing them in a general photon map, followed by the emission of a large number of photons specifically aimed at refractive objects. Photon hits from the high-density emissions are stored in a separate photon map so that the higher resolution data can be used to render caustics at a finer level of detail. Our project focuses primarily on photon mapping, and we will be basing our approach largely on Jenson's work.

A later paper published in 2008 by Sun et al. proposes a more complex version of this two-step implementation. This approach involves emitting photons into a voxelized scene and storing radiance distribution within voxels. The rendering step consists of tracing rays through the voxelized scene and summing the radiance from each voxel that reaches the eye. In addition to accurately capturing caustics, this method yields an efficient GPU implementation capable of running at interactive frame rates.

3 Implementation

Our proposed implementation is based largely on existing work and involves a two-step process. The first step, known as the emission step, involves casting photons from each light source into the scene and storing the illumination data in a photon map. Because we are limiting our project to rendering static scenes, in which objects and light sources do not move, the emission step will be performed once as a pre-computation. The second step, called the rendering or gathering step, involves rendering the scene using a reverse ray-tracing technique.

3.1 Photon Emission

Photons are emitted from light sources within the scene. Currently implemented are point lights sources, which emit light in all directions into the scene. For each light, we are building a projection map to help efficiently emit photons towards the geometry. The projection map represents a sphere surrounding the point light that is partitioned into cells of equal surface area. The scene is then projected onto this sphere, and any cells that contain scene geometry are then marked as such. Later, when emitting photons, this projection map can then be used to efficiently pick direction in which to randomly fire the photon, instead of wasting time emitting photons towards empty space.

For each photon, a ray is traced into the scene, and the intersection point is computed and stored in a photon map. Depending on the properties of the material, the photon will either be reflected or absorbed. We accomplish this by using a Russian roulette technique.

Normally, a photon would hit off an surface, losing some of its flux and then continue its reflection. A surface with a reflectivity of 0.5 would reflect photons at half their power back out. However, with Russian roulette, we tell half of the incoming photons to reflect at full power, which over time will stochastically model the same phenomenon.

If the photon is picked to be absorbed, it terminates its path and adds the hit into the photon map. However, if it is destined to be reflected, another Russian roulette is checked for whether it will diffusely or specularly reflected. Regardless of which way it will be reflected, if the surface is non-specular, the photon hit is also

*e-mail: lie2@rpi.edu
†e-mail: dolcea@rpi.edu

registered into the photon map. Thus, a single photon will leave multiple hits within the scene.

3.2 Rendering

In photon mapping, the rendering step typically involves tracing a ray from the eye into the scene through each pixel, recursively casting addition rays to handle reflection and refraction based on surface properties. In order to determine the radiance at a surface point, the photon map is searched for the nearest neighboring photon, and the photon data is used to approximate the incoming illumination at the surface point. Special care must be taken to ensure that each nearby photon is a valid contributor to the radiance of the surface point. For example, given a point on the surface of a thin wall, photons on the opposite side might be marked as contributors, thereby allowing light to leak unnaturally through the wall.

We must regretfully admit that, in its current state, the rendering step for our system is only partially implemented. However, it should be noted that we were able to lay a foundation for future improvement. The system is capable of rendering a scene via a simple raytracer, but unfortunately it does not yet support reflected or refracted rays. This limits the final rendering in that it does not allow for the accurate rendering of reflective or refractive surfaces.

Despite these drawbacks, the rendering of diffuse surfaces is computed using the nearest neighbor search algoritm as follows. For each pixel, a ray is cast from the eye into the scene, and the closest intersection point is found. The nearest neighbor search is then conducted for N photons by incrementally expanding a sphere centered at the surface point until the sphere contains a specified minumum number of photons. Although the starting radius of the sphere can vary depending on the size and photon density of the scene, our system currently begins with a radius of 0.1. The sphere is then recursively tested for intersection with the bounding volumes of the photon map, until all contained photons are found. Of the contained photons, the nearest N are determined by comparing squared distances. If the sphere contains fewer than N photons, the sphere radius is doubled and the intersection test is recomputed. This step is repeated incrementally until enough photons are found or until the radius exceeds a specified cutoff threshold. For our purposes, the radius was restricted to a maximum of 0.4.

Once the nearest N neighboring photons have been located, their radiance contributions are summed. Each contribution is determined based on the photon's RGB power value and incident direction in accordance with the local BRDF of the surface. Currently the rendering step supports only diffuse Lambertian surfaces, for which the BRDF can be quickly evaluated by taking the dot product of the photon direction and the surface normal. The sum of the radiance values is then divided by an estimate of the local photon density based on the distance to the farthest of the N neighbors.

3.3 Photon Map Implementation

Our photon map is implemented using an octree data structure, which represents a hierarchy of bounding volumes. Each node of the tree represents a three-dimensional bounding box. An internal node signifies that the bounding box has been evenly divided further into eight smaller boxes, each of which is represented by a child node. Each leaf node stores the set of photons contained in its bounding box. If a leaf nodes set exceeds a specified threshold, its bounding box is divided by expanding the node into eight child nodes and by storing each photon in the appropriate child.

The octree data structure allows the system to more efficiently find the neighboring photons for each point during the render step. The nearest neighbor search is comprised of an intersection test in which the bounding volumes of the octree are compared with a specified sphere. The search begins at the root and continues down the tree recursively for any child volumes that intersect the sphere. At each encountered leaf node, the photons stored are tested for intersection individually.

4 Results

We have implemented the two-pass photon map model and in this section we will present our initial results from this implementation. All simulations were done on a 2.0 GHz Intel Core Duo T2500 running both Linux and Windows. We created two separate visualizations, each of which is 600 pixels wide.

The below table displays the memory consumption of the octree structure for renderings of the same scene across a variable number of emitted photons and for different octree threshold values.

		Octree Threshold		
		10	100	1000
Emitted Photons	5k	0.5 MB	0.4 MB	0.4 MB
	50k	4.9 MB	4.1 MB	3.9 MB
	500k	50.3 MB	40.6 MB	39.8 MB

The left visualization shows the actual photon map used to generate the image on the right visualization. Because the power of each individual photon is very low we have scaled up their power uniformly just for the photon visualization, so that we can examine the photon emission. We can see the color bleeding effect of diffuse surfaces that are nearby, such as in the Cornell box simulation.

The right visualizations show that actual image produced by the rendering step. Clearly the accompanying images are marred by a large number of unwanted artifacts, most notably the large amount of noise in the image, which results in a speckled distribution of bright and dark spots. This is a side-effect of the incomplete rendering step which is not yet capable of properly blending the contributions of photons. We hope to improve our algorithm in order to achieve greater accuracy and reduced noise, but unfortunately we have run out of time.

5 Conclusion

It is difficult to draw solid conclusions from our incomplete results. Despite this, we are satisfied with the emission step algorithm in its ability to accurately and evenly distribute photons within the scene. Given additional time, we would we able to more fully complete our rendering step in order to better validate our results.

References

I. Ihrke, G. Ziegler, A. Tevs, C. Theobalt, M. Magnor, H.-P. Seidel, "Eikonal Rendering: Efficient Light Transport in Refractive Objects" , ACM Trans. on Graphics (Siggraph'07), 2007, to appear.

Sun, X., Zhou, K., Stollnitz, E., Shi, J., and Guo, B. 2008. Interactive relighting of dynamic refractive objects. In ACM SIGGRAPH 2008 Papers (Los Angeles, California, August 11 - 15, 2008). SIGGRAPH '08. ACM, New York, NY, 1-9.

Henrik Wann Jensen: "Global Illumination using Photon Maps". In "Rendering Techniques '96". Eds. X. Pueyo and P. Schrder. Springer-Verlag, pp. 21-30, 1996.