Rendering Caustics Without Photon Maps

Kate Kelly [kellyk3]

Paul DiPastina [dipasp]

Rensselaer Polytechnic Institute [@rpi.edu]

ABSTRACT

Despite Photon Mapping being a popular and widely used method for rendering certain aspects of light that are un-raytraceable, ie caustics or refractive properties, we set out to discover alternate methods to create caustic effects. We present a very limited-scope algorithm that allows caustics to be integrated into a raytracer for certain spherical or lens-like geometries; we also present a real-time solution using GLSL and pixel shading.

1. INTRODUCTION

1.1 MOTIVATION

We were interested in what other methods existed for applying caustics to a scene, outside of Henrik Wann Jensen's photon mapping technique taught in class. In particular, we wanted to discover if there existed an algorithm to integrate caustics, which ordinarily must be traced from the light, smoothly into a standard backwards-raytracing (traced from the eye) model. In addition, we discovered a method to simulate real-time caustics without raytracing that still adds a large amount of realism to a scene. Initially, our focus was solely to attempt to implement the first method, but when that utterly failed, we required something solid to have results for our paper – and so we expanded our area to include real-time caustics.

1.2 BACKGROUND

In Masa Inakage's 1986 paper, "Caustics and specular reflection models for spherical objects and lenses," he solves the problem of using an eye-traced ray to create caustics, by having knowledge of the refractive medium. He creates a cone-shape to model the convergence of parallel light rays passing through a lens and converging on its focal point. The major idea behind that paper is that the ratio of areas matches the ratio of light – that the light passing into the lens becomes compressed and intensified into the area smaller than the lens (if nearer than the focal point) or spread and dimmed if the surface the caustic is being projected onto is further than that.

In 1990, "A Survey of Shadow Algorithms" was published, a compilation of work from Woo, Poulin, and Fournier. In it, they mention the Inakage paper and its flaws. Inakage's method is only valid for collimated (a parallel beam of) light rays, when ordinarily a point light would be used. They criticize the algorithm more by pointing out that it is only valid for a single, somewhat thin lens. Large spheres will not have the proper patterns of convergence.

Shah, Konttinen, and Pattanaik created a paper in 2005 on "Caustics Mapping: An Image-space Technique for Real-time Caustics." This paper was one of a slew of publications at the time to take advantage of the advancements in GPU hardware and GPU programming that allowed better real-time performance by using all the bandwidth of the pipeline instead of using expensive precomputed calculations like raytracing. Like shadow mapping, Caustics Mapping deviates from the standard of geometry-space and instead creates an image-space map of the caustic from the point of view of the light. This map is then put into a texture, which is then applied onto the "receiver surfaces."

2. IMPLEMENTATION

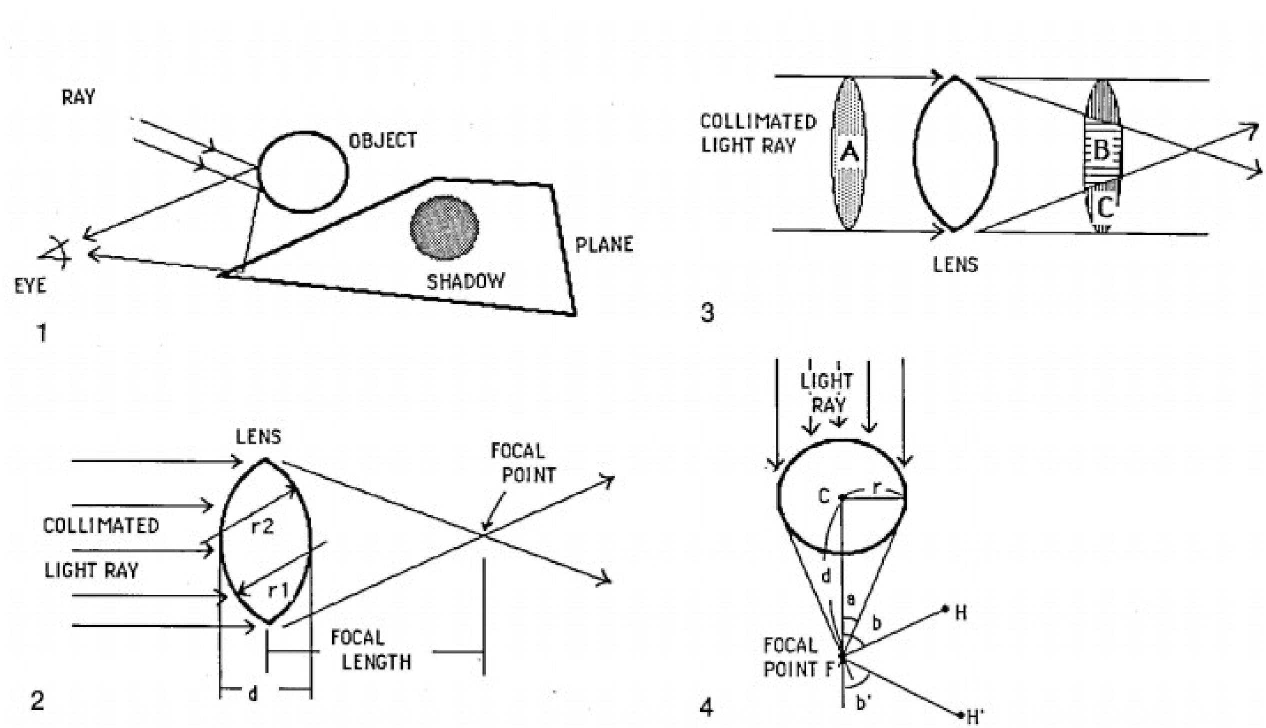
Inakage's paper begins with the Lens Makers' Formula, which states that "a reflective object is merely a special case of a refractive object," with the exception that the refraction index is negative.

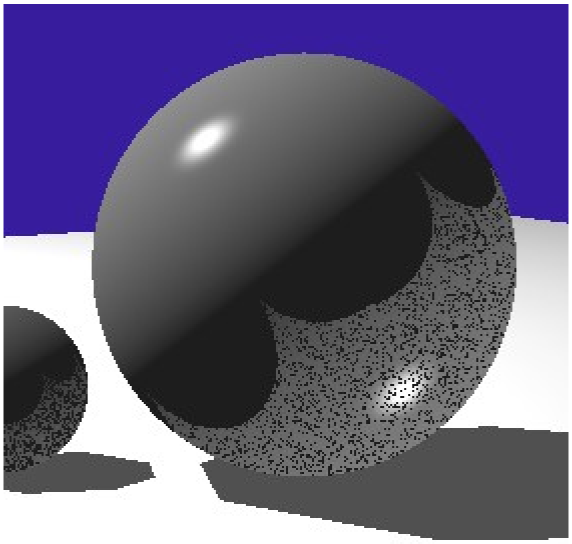
$$1/f = (n1 - n2)*(1/r1 - 1/r2 + ((n1 - n2)*d)/(n1*r1*r2))$$

r1 and r2 are the radii of curvature, d is the thickness of the lens (for simple lenses this is 0), n1 is the refraction index of the material, passed in through the .obj file, and n2 is the outside refractive index (usually 1 for air).

The figure below shows the specific circumstances required for this algorithm to work – namely, the parallel light rays, which is the major drawback of this algorithm.

Caustics with Inakage's algorithm are done by adopting Snell's law – when a light ray passes through a lens it gets refracted in a certain way, and if we know a ray is going to pass through a refractive material with certain geometry, we can perform simple algebra and create a caustic at the ray-traced point. Below: The sum of the intensity in area A is equal to the same sum in area B, ergo since B is a smaller area, it will be brighter – this is a caustic. On the next page you will see a picture of how attempting to implement this algorithm turned out.





As you can see, the initial attempt to implement Inakage's algorithms did not meet with success. We turned to the real-time caustics algorithm described by Shah, Konttinen, and Pattanaik next. Paul was in charge of translating the HLSL/DirectX code into GLSL for use with our program, but at this point in time it is not finished. Hopefully we will have a working demo for Wednesday's presentations.

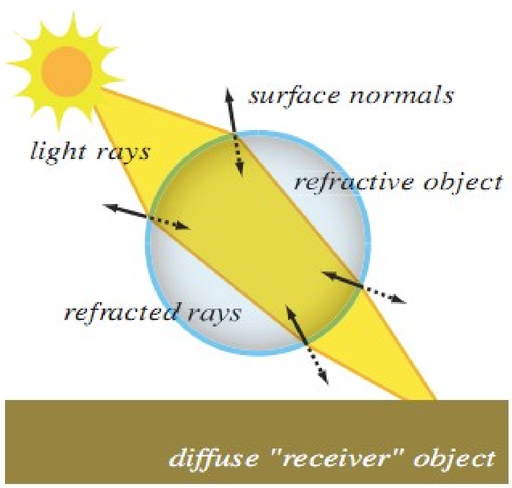


Fig. 3. Diagram showing how multiple light rays can refract through an object and converge at the same point on a diffuse surface.

How the real-time caustics are implemented:

First, obtain the 3D coordinates of the geometry that is going to have the caustics mapped onto it. Next, get the 3D coordinates and surface normals of the refractive object – this would be done using an *.obj parser that knows how to handle indexes of refraction. Next, render the refractive object to a texture from the point of view of the light. This is the texture that will be projected onto the receiver surfaces. This is created by "splatting" points from each vertex of the refractive grid along the direction of the refracted light. Then, an optional shadow map algorithm is computed for added realism. Finally, render the scene with the caustic texture layered over the receiver geometry.

REFERENCES

Inakage, Masa 1984. "Caustics and specular reflection models for spherical objects and lenses." The Visual Computer, Vol 2, Issue 6.

Woo, Poulin, and Fournier. 1990. "A Survey of Shadow Algorithms." Dept of Computer Science, University of Toronto.

Shah, Konttinen, and Pattanaik. 2005. "Caustics Mapping: An Image-space Technique for Real-time Caustics." Technical Report, School of Engineering and Computer Science, University of Central Florida, CS TR 50-07, 07/29/2005 (Submitted for Publication).