Light Refraction with Dispersion

Steven Cropp & Eric Zhang

1. Abstract

In this paper we present ray tracing and photon mapping methods to model light refraction, caustics, and multi colored light dispersion. We cover the solution to multiple common implementation difficulties, and mention ways to extend implementations in the future.

2. Introduction

Refraction is when light rays bend at the boundaries of two different mediums. The amount light bends depends on the change in the speed of the light between the two mediums, which itself is dependent on a property of the materials. Every material has an index of refraction, a dimensionless number that describes how light (or any radiation) travels through that material. For example, water has an index of refraction of 1.33, which means light travels 33 percent slower in water than in a vacuum (which has the defining index of refraction of 1). Refraction is described by Snell's Law, which calculates the new direction of the light ray based on the ray's initial direction, as well as the two mediums it is exiting and entering.

Chromatic dispersion occurs when a material's index of refraction is not a constant, but a function of the wavelength of light. This causes different wavelengths, or colors, to refract at slightly different angles even when crossing the same material border. This slight variation splits the light into bands based on wavelength, creating a rainbow. This phenomena can be recreated by using a Cauchy approximation of an object's index of refraction.

3. Related Work

A "composite spectral model" was proposed by Sun et al. [2000], which decomposed the light into a smooth component and a collection of "spikes". They use the same general method as regular ray tracing; rays are cast from the eye and the shading is determined by the contribution of the light at the rays termination point. What they have added is the branching of polychromatic rays into multiple rays of different wavelengths.

Weildlich and Wilkie [2009] have also proposed various ways to handle dispersing polychromatic light through various colored mediums. In their ray based implementation, they explore the refraction of polychromatic light in multiple colored gems, different surfaces, and prisms.

Finally, Mihai and Strajescu [2007] proposed a way to convert wavelength to RGB color. Since there is not a single RGB triple for each wavelength, we can only approximate a range for the color to appear. Therefore, the writers proposed approximate ranges for each color, and given the wavelength, return a certain RGB set.

4. Basic Refraction

4.1 Snell's Law

Snell's Law, also known as the Snell-Decartes law, describes the relationship between the incident angle and the outgoing angle of a light ray traveling across the border of two mediums. Snell's law can be used to calculate the new direction of a ray based on the inherent properties of the mediums (the indices of refraction), and the angle of incidence of the ray. [1]

$$\frac{\sin(\theta_1)}{\sin(\theta_2)} = \frac{\eta_2}{\eta_1}$$

Fig. 1 This formula states that the ratio of the indices of refraction is also the ratio of the sines of the angles of incidence. Here '1' is the medium the ray is leaving and '2' is the medium the ray is entering.

4.2 Implementation

In order to model refraction in ray tracing an object's refractive color and refractive index must be included in the mesh object data structure. We assumed that the refractive index of all space outside of objects is exactly 1, that of air or a vacuum. All ray intersections with refractive objects can be considered either entering the object or exiting the object. Thus the ratio of refractive indices is either the intersected object's refractive index, or the inverse of its refractive index, respectively.

For the ray tracer to determine if a ray is entering or exiting an object, it is important to consistently handle the normal that is returned when a ray intersection occurs. By having objects always return a normal that is pointing outside of, or away from the object, the dot product of the ray and the normal can be used to distinguish between a ray entering or a ray exiting an object. A positive dot product means the ray is traveling in the direction of the normal, and must have collided as it tried to exit the object, while a negative dot product means the opposite, and the ray was on course to enter the object.

When an object is both slightly reflective and refractive, traced rays branch in both the reflective direction and the refractive direction every intersection. This is consistent with the physical world. The return value of such a ray is calculated as the weighted average of the reflected and refracted traced ray return values. The weights of the returned ray colors are based on the object's properties, the magnitudes of the inherent reflected and refracted colors respectively.

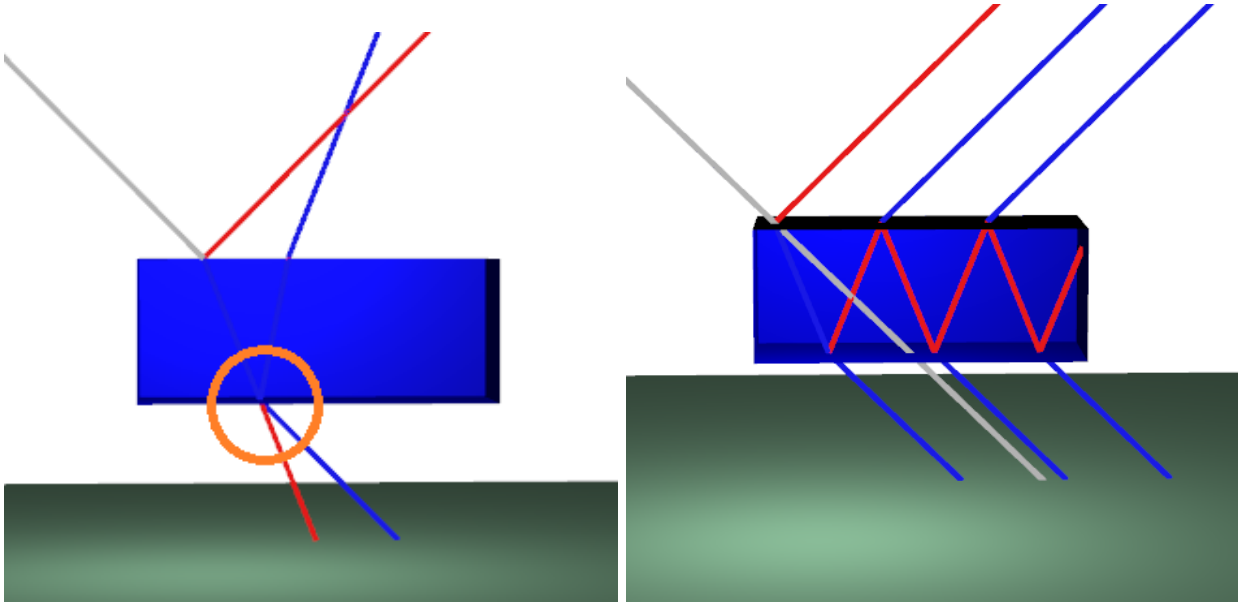
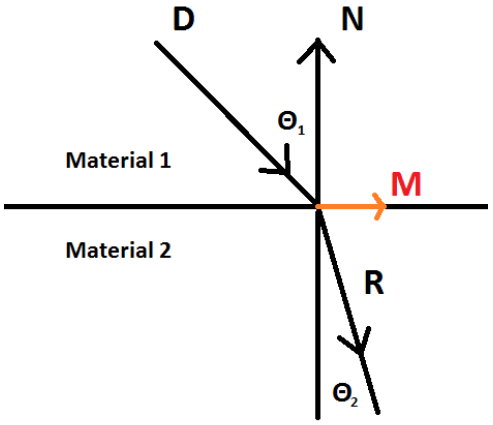


Fig. 4.2: The above images show traced refractive rays in blue and reflected rays in red. The left image shows an incorrect handling or normals, where refractive and reflective rays bounce incorrectly (see the intersection inside the orange ring). The right image shows a correct handling.

An epsilon error value was used along with the intersected object's normal to guarantee that a calculated float position was inside or outside of an object. This ensures that the new ray begins on the side of the surface that avoids another collision at the same spot. The desired start location of a ray depended on whether the ray was a reflection or a refraction ray, and if it was entering or exiting an object.

4.3 Calculating the Refracted Direction

The new direction of the ray is calculated using the following formula:



$$M = D + N\cos(\theta_1)$$

$$R = M\sin(\theta_2) - N\cos(\theta_2)$$

Where D is the incoming ray, N is the Normal of the area of the material hit, M is the direction perpendicular to the normal and away from the angle of incidence, and R is the refracted ray. Before this calculation however, an important check must be done to properly handle total internal reflection.

4.4 Total Internal Reflection

Snell's Law can be rewritten as the following:

$$\sin(\theta_1) = \frac{\eta_2}{\eta_1}\sin(\theta_2)$$

This form shows that there exists a problem when $\sin(\theta_1) > \frac{\eta_2}{\eta_1}$. In these cases the value of $\sin(\theta_1)$ must be greater than 1, for which there is no existing θ_2. These cases are known as total internal reflection, as the light rays are unable to refract across the medium boundary, but are still able to reflect off of the boundary.

To handle these cases, a simple check is done before attempting to calculate θ2. When total internal reflection is detected, the ray is only reflected. In cases where an object has no reflective component (its reflective color is black) the refractive color is used in reflection.

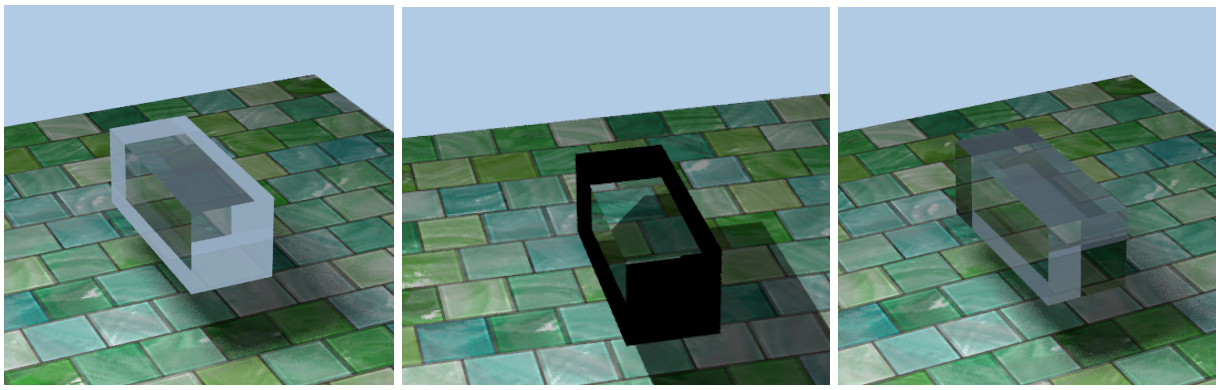


Fig. 4.4.1: The left image shows a prism rendering with no check for total internal reflection handling. The middle image shows correct total internal reflection handling but without the fix to avoid non-reflective color reflection. The right image shows the most correct rendering of refraction with internal reflections.

4.5 Photon Mapping

The largest difference between handling refraction in ray tracing and handling refraction in photon mapping is in the handling of branching paths. Photons have the potential to reflect or refract in many cases, so a russian roulette method was used in order to avoid storing more photons than are initially shot from light sources. A photon intersecting an object randomly chooses to either bounce diffusely, reflectively, or refractively based on the magnitude of the

intersected object's diffuse, reflective, and refractive colors, respectively. This russian roulette approach resulted in accurate caustic generation.

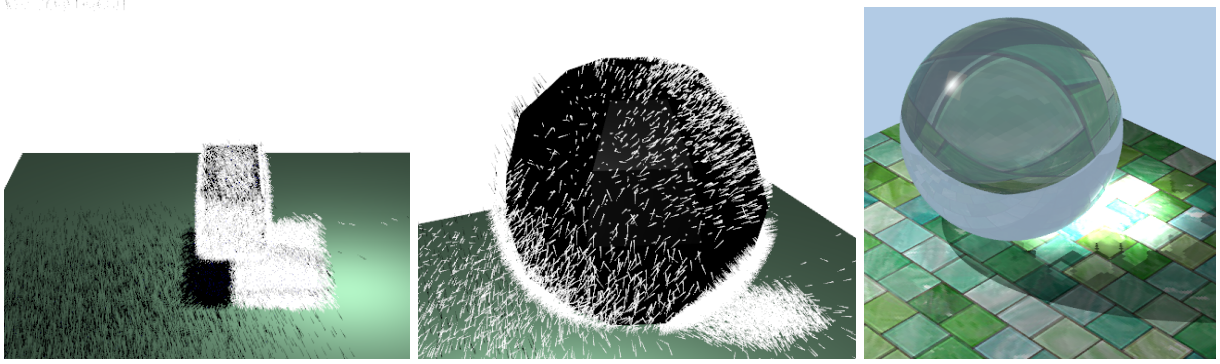


Fig. 4.5.1: The left shows caustics created by light refracting through a prism (the black photons in the prism picture are photons that went through total internal reflection). The middle shows light focused through a sphere, while the right shows a rendering of the sphere caustics.

5. Colors

5.1 Adding Wavelength

Adding dispersion effects to photon mapping required adding a wavelength factor to the refraction calculation, as well as a wavelength property to the photon class. To emit photons with different wavelengths we originally considered emitting 'white' light photons from the light source, and splitting them into multiple monochromatic photons of unique frequencies. However, we decided to instead emit all photons as monochromatic with a randomly generated wavelength. This avoids the need to track a photon's monochromatic state, but on average requires more photons to be shot in order to render realistic scenes.

Research was done on generating RGB values from wavelength in an attempt to create a more continuous spectrum, but such functions are non-trivial. A large portion of the difficulty of those conversions is that there is not a one to one correspondence between rgb values and wavelength. Multiple combinations of red, green, and blue can create the same wavelength of color.

Therefore, when calculating the color given a wavelength, we can only give a rough estimate of what the color should be. In our implementation we approximated colors given a wavelength and linear RGB function (ex. 645-780 is red). We allowed for color interpolation when rendering the photon mapping to help blend the colors that we did not cover with our estimations.

5.2 Cauchy's Equation

We used Cauchy's Equation for refractive indices, modeling the dispersion as a cauchy sequence.

$$n(\lambda) = B + \frac{C}{\lambda^2}$$

The B is the object's given index of refraction, and C is an arbitrary constant based on real world materials. We chose a material similar to crown glass, so our B and C values are roughly 1.5 and 4.2, respectively. We calculate a 'new' set of indices of refraction for each wavelength because each wavelength travels through the space slightly differently. We use the original index of refraction as the base and modify it per photon to get different angles of refraction.

5.3 Calculating New Direction

The new direction of the photon is handled similarly to previous refraction. However, now instead of having constant value for the indices of refraction for the air and the objects, each photon essentially travels through a different refractive index based on its wavelength. A straightforward way of thinking about this would be to calculate how fast the photon travels based on the wavelength alone, but by changing the refractive index per photon, we create the same effect with less costly computations. By treating each photon like it is traveling through different mediums, we get variations in angles even when they refract with the same objects.

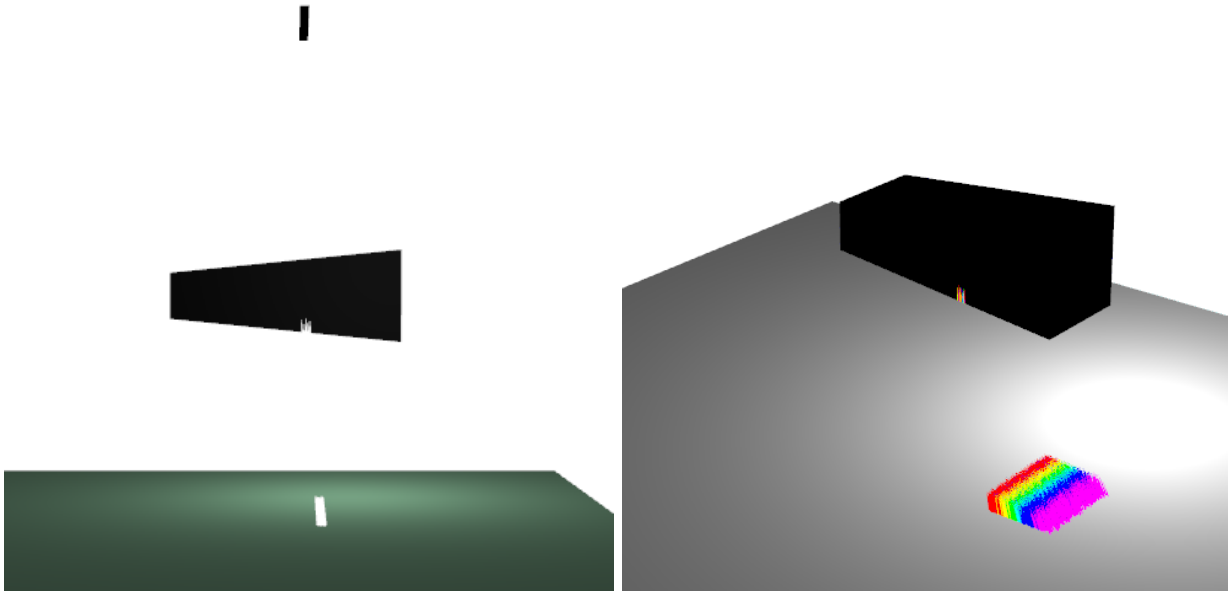


Fig. 5.3.1: The left image shows the photon mapping without wavelengths, the scene casts rays in a straight line, with diffuse bouncing turned off in order to keep the scene clean. The right image shows the same scene after the Cauchy

5.4 Rendering/Calculating Colors

Each photon is assigned a color based on its random wavelength. We approximated the ranges for certain colors, and left it up to the interpolation while photon gathering to create the intermediate colors we did not cover.

We calculated the color using the method outlined by Mihai and Strajescu. However, their exact model was giving us some funny colors so we ended up modifying it.

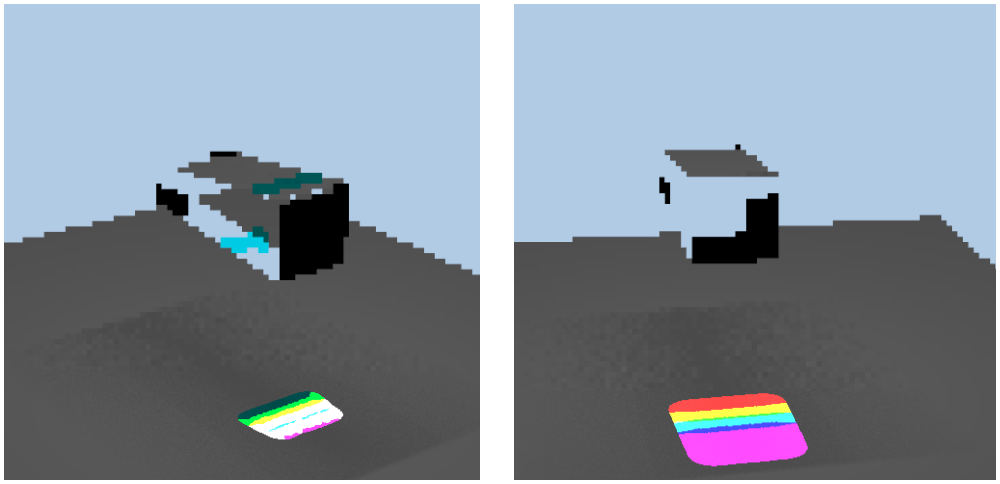


Fig. 5.4.1: The right image is the original colors from the paper implementation, the left is the colors from our modified values

We replaced the energy value to instead store a value for color, and so when we do photon gathering to gather intensities, we also gather colors. In theory, this should allow for nice color interpolation between photons of different color to create the intermediate colors we did not code for.

Due to our photon gathering, some trouble came up while rending colors, as sometimes the colors would interpolate together too much and become splotches of white light. This is mostly due to the search radius each area is given to search for photons. If the area is too large then the colors averaged together become muddled. If the area is too small, there is no interpolation, or not enough photons will be found.

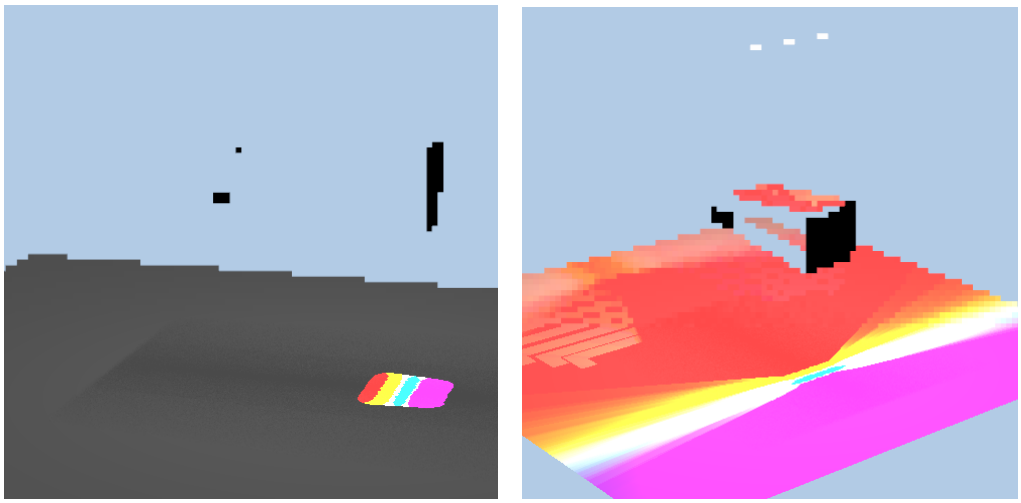


Fig. 5.4.2: The left image is an example of white light interpolation, the right is an example of incorrect interpolation when photon gathering

6. Limitations

One limitation is with the assumption that a ray can only ever interface between an object and air. This method can not refract rays between two mediums exactly next to each other. If two objects with non-next to each other are less than epsilon apart then our implementation will not be able to calculate the refraction that occurs as the ray passes into the 2nd object. This is because when the ray exits the first object, it expects to be put outside of the object by some epsilon, into a vacuum. If the closest object is less than epsilon away, then the ray will be put into the new object without collision. The ray or photon would then refract as it leaves the second object, having never refracted to enter.

7. Results/Discussion

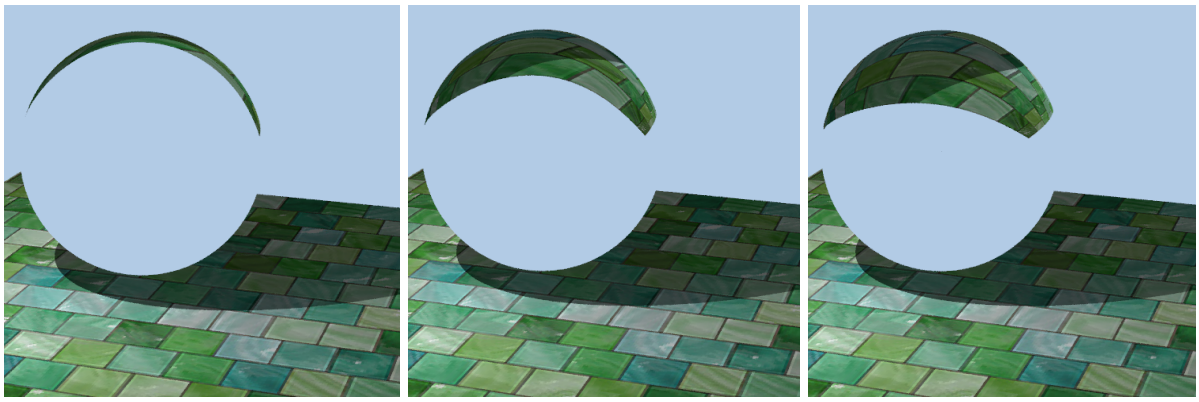


Fig. 7.1: This scene is of perfectly refractive spheres with no reflection component. The index of refractions are the only differences, beginning with 1.2 on the left, 1.4 in the middle, and 1.6 on the right.

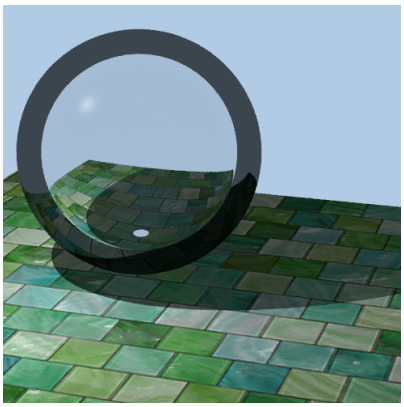


Fig. 7.2: This scene shows a reflective sphere with a physically impossible index of refraction of 0.8. It is roughly equivalent to looking at an air bubble while underwater, and the dark ring represents areas where total internal reflection is taking place.

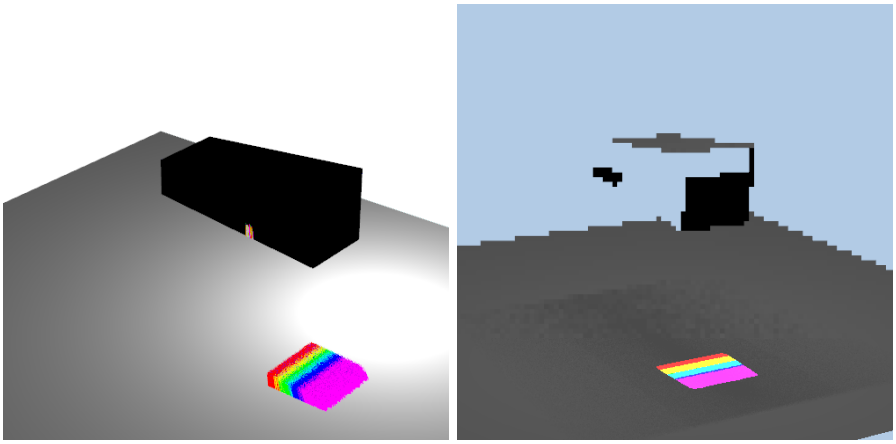


Fig. 7.3: On the left is an image with the photons mapped only. On the right is an incomplete photon mapping render of the light.

8. Future Work/Conclusion

The first thing we would like to improve is our photon mapping rendering. It is a little inconsistent and it has some flaws when we put too many constraints on it. Most notably we would like to have it more accurately handle colors.

In today's modern computing society, speed is becoming more important with man work hours becoming more important than computer hours. Therefore, we would like to parallelize our algorithm to speed up rendering times.

Additionally, another more global type of dispersion is Raleigh Scattering. The most well-known representation of this is sunsets. We would like to implement this sort of global light dispersion in a future iteration.

One of our original ideas was also to do internal refraction with gems and crystals. If possible we would like to revisit that idea in the future, and refine our implementation to handle gems.

9. References

[1] Weidlich, A. and Wilkie, A. (2009), Anomalous Dispersion in Predictive Rendering. Computer Graphics Forum, 28: 1065–1072. doi: 10.1111/j.1467-8659.2009.01483.x

[2] Sun, Y., Fracchia, F. D., and Drew, M. S. 2000b. Rendering light dispersion with a composite spectral model. In Proceedings of the 1st International Conference on Color in Graphics and Image Processing. 51--56.

[3] Dragoş Mihai, Eugen Strajescu (2007), From Wavelength to RGB Filter. U.P.B. Sci. Bull., Series D, Vol. 69, No. 2, 2007

10. Wrapping Up

Rough Time to Completion: ~35 hours

Steven's Contributions: Built many scenes, added refraction to mesh data structure and parser. Did most of normal/ray tracing refraction debugging (epsilon and normal handling)

Eric's Contributions: Initial refraction computations, added colors to refractions, wavelength to RGB, photon mapping with colors

Overall we typically worked side by side for the whole project, we were both knowledgeable on each other's parts.