Rendering Diamond Like Structures with Ray Tracing and Photon Mapping

CS 580 Project Report

Ningfeng Huang*
USC
ningfenh@usc.edu

Mohsin Ali
USC
mohsinal@usc.edu

Hassan Nawaz
USC
hnawaz@usc.edu

Joy Dsouza
USC
jwdsouza@usc.edu

ABSTRACT

In this project we implement ray tracing with photon mapping to render diamond like structures. The important contributions of the project can be divided into five major phases, ray tracing diffusive surfaces, adding fresnel reflection and refraction for transparent materials, adding soft shadows, implementation of photon mapping for caustic effects and finally re using photon mapping for global illumination. Then we went on to model two different scenes using our implementation, an 8 facet prism placed inside a cornell box and a 360 degree view of a realistic diamond with 153 faces. Initial implementation was completed in MATLAB without using any built in algorithms and later on a faster version of ray tracing was implemented in C++ for rendering a 360 degree view. A video showing 360 degree view of the diamond along with the github code can be found on our webpage (www-bcf.usc.edu).

Keywords

580 Course Project, Computer Graphics, Rendering, Photon Mapping, Ray Tracing

1. INTRODUCTION

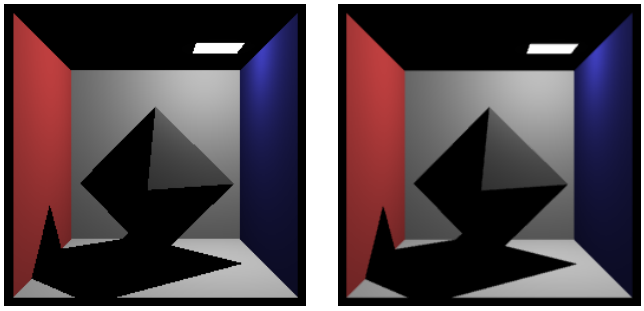
Raytracing is a technique for rendering images by tracing the path of the light. In contrast to the real world where light hits an object and enters the camera, this technique sends out rays from the camera and see what objects they encounter. This ray is reflected once it hits a shiny surface and continues in this direction until it encounters any other object. The object it then hits also gets a reflected color from the initial object so can see reflected images of one object in another adding to the realistic effects. Similarly, depending on the object, a ray can also be refracted according to the angle of refraction into the object. If a ray hits an object before hitting a light, that part of the object is said to be in a shadow.

Figure 1: The final rendering result of this project: a brilliant cut diamond with 153 facets sitting on a gray diffusive table.

Although ray tracing is computationally expensive it still has multiple advantages such as handling shadows, multiple reflections and refractions. Shadows are important because they make the lighting much more realistic. Alternate approaches like shadow maps do not scale well because in some cases such as movie scenes in Disneys Cars, there are thousands of lights and this can lead to asset management problem [1]. Using ray tracing solves this problem. Ray tracing is also used for ambient occlusion which measures how much light is reached to a point. This is used in movie production as well as an indicator for proximity of other objects. Generally Ray tracing produces highly realistic images as compared to their counterparts rendering algorithms.

There is also some competetion from other rendering algorithms like rasterization based systems that draw the the triangles in a sequential manner and handle overlaps in dif-

*The code for this project is openly available hosted at github www.github.com. A 360 degree view of the rendered diamond can be access on our webpage www-bcf.usc.edu



(a) (b)

Figure 2: Basic ray tracing rendering of a diffusive eight-facet prism. (a) without anti-aliasing and (b) with anti-aliasing.

ferent objects by using depth maps. A later triangle can overwrite an existing triangle if its closer. In contrast to this, ray tracer maintains the triangles in a Data Structure which is checked for intersection with each ray. Although this makes the ray tracer slow but its helps to give better quality results for metal, glass and water like materials. Ray tracing in more suitable for applications where realistic images are required ahead of time as compared to real time rendering.

Ray Tracing on itself is good but not brilliant. The main limitation of ray tracing is that the objects are isolated with each other. This means when a ray intersects with one triangle it doesn't have any information of the surrounding environment. In order to add to the realism of the scene we need to add photon maping. In photon mapping photons are emitted from the light source which act as small packets containing energy. They then collide with different geometry and then either get absorbed or reflected to hit other geometry. After that ray tracing gather these photons that collided and were stored in photon maps and compute the color of that region accordingly. One of the things simple ray tracing lacks is the ability to give caustic effects. Caustic effects are patterns made by the reflected or refracted light after bouncing off from specular surfaces and creating patches of concentrated light on a diffusive surface. We need to trace the paths of each and every photon that hits the surface if we want to get this effect making photon mapping the ideal candidate. In addition, global illumination is needed to add to the realism of 3D scenes where not only the light from lights source contributes to the final color but also the reflected and refracted light counts which is also referred to as indirect illumination. Using photon mapping with raytracing adds to the accuracy of the rendered scene.

2. IMPLEMENTATION

We divided our implementation into five major milestones. We first start of with basic ray tracing of a diffusive surface, moving on to handling Fresnel reflections and refractions in the next step. Our third milestone is to achieve soft shadows and then we move on to implementing caustic effects with the help of photon mapping. Finally we implement global illumination using a global photon map. Figure 1 shows the final rendering result of our project. It is a brilliant cut diamond with 153 facets sitting on a gray diffusive table. In this figure, one can observe the following features. (1) The shading of the diamond surface is completely from the multiple reflection and refraction of the light. The diamond itself doesn't have any color. (2) The shadow of the diamond is soft. (3) Scattered light from the diamond can be observed on the diffusive table surface. (4) The scene is purely in gray scale, however, some area on the diamond and the table surface appear to be colorful due to the dispersion of the light. (5) The scene is anti-aliased.

2.1 Basic Ray Tracing

To clearly demonstrate our approach in this project, we show here a simpler scene consists of a eight-facet prism in a Cornell Box. As you can see in Fig 2 (a) we have rendered a prism with just basic ray tracing that only caters for diffusive surfaces. The idea is to create light rays which originate from the camera position and pass through each pixel of the screen. This maps to a call to raytrace function for each pixel, inside which we trace the traversal of light in the scene and return a color which is then flushed onto the pixel.

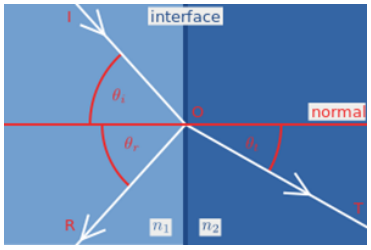
In the raytrace function, we select the nearest triangle from the set of all triangles and check the intersection with it. The triangles can be of two types i.e. light sources or diffusive. The intersection is checked among all the triangles in the geometry. We compute the normal for each triangle so we get the plane equation and we already know the direction of light ray. As a result, intersection problem boils down to mathematical problem of determining the intersection of a line with a plane. In case of multiple intersections we choose the nearest triangle.

Once we have the intersecting triangle, we calculate the shadow ray for each light source and check whether there is some other triangle which comes in the way of this shadow ray before it reaches the light source or not. We then sum the colors from shadow rays of all light sources that are not blocked by any other triangle. So, depending on the number of shadow rays intercepted, we determine if there is a shadow and how dark it is. Otherwise if the intersecting triangle was the light itself, we return the color of light source. In our implementation, prism is actually supposed to be transparent but the in this milestone we do not support transparent surfaces.

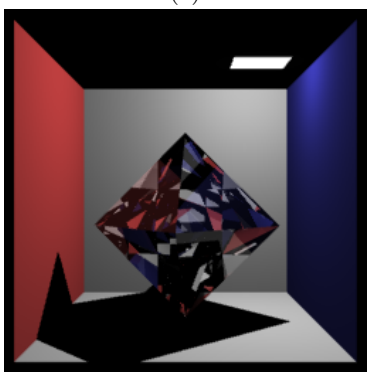
Fig 2 (b) shows the same rendering but with the anti-aliasing. Anti-aliasing(super sampling) is natural to implement in ray tracing algorithm. The camera censor (direction of the rays) is shifted by subpixel distance towards multiple directions and the rendering results are summed up by certain weighting coefficients.

2.2 Reflection and Refraction

Since the final goal of this project is to render a diamond, we now add support for transparent surfaces. Now if our intersecting triangle is transparent the light would either be reflected back or refract and change mediums (Fig 3 (a)). The decision depends on the direction of light, surface normal and incident medium (air or diamond in our case). For the physics behind this we use the SnellâĂŹs formula (equation 1, 2) to calculate angle of reflection and refraction and Fresnel's equation (equation 3, 4, 5) to calculate the ratio of

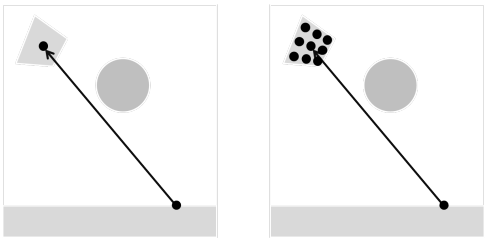


(a)

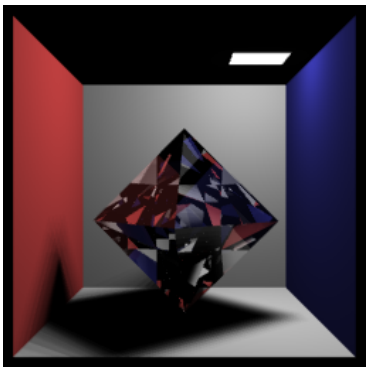


(b)

Figure 3: Adding reflection and refraction of the light at a transparent interface. (a) Snell's law of reflection and refraction. (b) Rendering of the transparent prism with the refractive index of 2.4.



(a)



(b)

Figure 4: Implementation of the soft shadow. (a) Basic principle of the area light source. (b) The rendering result of the prism scene with the soft shadow.

reflection and refraction. The reflection (transmission) coefficients will depend on the polarization of incident light. To mimic the natural light which is normally unpolarized, the average of two reflection coefficients is taken. In case where a ray moves from air to diamond, we calculate directions of reflected and refracted rays and call raytrace function on them recursively. Note that each ray is divided into two for reflection and refraction. Once a color is returned by this raytrace function for reflected ray, it is then multiplied by the reflection ratio determined by Fresnel and similarly the refracted ray is multiplied by the ratio entered into the diamond. In Fig 3 (b) you can notice that different faces show different colors, that is because reflected rays from each face eventually end up on different walls, bringing back their colors.

In case where light is travelling from diamond to air, if Fresnel formula will indicate that total internal reflection has occurred so the ray would be reflected back into the diamondand will keep on bouncing back and forth, otherwise, the ray is divided into two parts one each for reflection and refraction. In our implementation, the limit of recursive calls for one ray is limited to 5 after which the ray is said to have faded out.

$$\theta_r = \theta_i \tag{1}$$

$$n_1 \sin\theta_i = n_2 \sin\theta_t \tag{2}$$

$$R_s = \left[\frac{n_1 * \cos\theta_i - n_2 * \cos\theta_t}{n_1 * \cos\theta_i + n_2 * \cos\theta_t}\right]^2 \tag{3}$$

$$R_p = \left[\frac{n_1 * \cos\theta_t - n_2 * \cos\theta_i}{n_1 * \cos\theta_t + n_2 * \cos\theta_i}\right]^2 \tag{4}$$

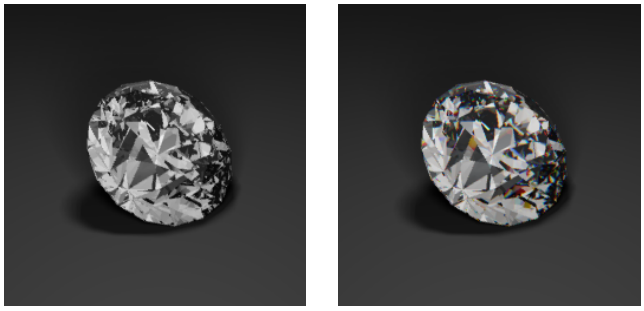
$$R = (R_s + R_p)/2 \tag{5}$$

2.3 Soft Shadows and Dispersion

If you notice figures 2 and 3, you would notice that our shadow has well defined boundaries. However, that does not seem realistic specially with the a square light and a diamond underneath it. In order to make it look more realistic we implement soft shadows.

The origin of soft shadow is the finite area (instead of infinitesimal) of the light sources. To achieve the soft shadow, instead of casting the shadow ray towards one point light source, we divide each square light source into 5×5 light sources spread through the physical area of the light. Then we cast multiple shadow rays to each of the sub-light source and count the number of rays being blocked by other objects.(Fig 3(a)) Note that in Fig 3(b), boundaries are very smooth so are the shadows.

Moving on to dispersion. The physics of dispersion is that for different color the refractive index of the material is different. Take diamond for example, the refractive index is 2.4 for red light, 2.43 for green light and 2.46 for blue light. As a consequence, the incident white light will be separated to rainbow like color. In our ray tracing code, instead of having just one ray tracer get all the colors for us we use 3 different



(a) (b)

Figure 5: Supporting the dispersive material.(a) Rendering without material dispersion. (b) Rendering with material dispersion.

parallel ray tracing instances with the corresponding refractive indices. Each ray tracer calculates either red,green or blue channel of the final image which are then merged at the end. This is done so that we can have different refractive indexes for different colors. The result can be seen clearly in Fig 5 by comparing the rendering with and without dispersion. You can note that even all the surface and light source is of grey scale, there are slight shades of different colors which are formed due to dispersion. Some principle has been used to render the caustic and global illumination which will be discussed in the next section. This results in the colorful scattering of the light on the table surface shown in Fig 1.

2.4 Photon Mapping

Since we are rendering a diamond, even such a soft shadow is extremely unrealistic because light can also pass through the diamond and also bounce off from the walls. In real life when light passes through materials like glass a caustic effect can be seen in the shadow where reflected light forms patterns on the floor. Light concentrates on some patches more than others. Modeling such effect is not possible with simple ray tracing. In ray tracing, all the light is absorbed by the first diffusive surface they land on but in photon mapping a diffusive surface can either absorb the photon or reflect it.

Photon mapping involves two passes. We first create a lot of photons by randomly choosing points on the light and random directions towards the scene. We then call photon mapper for all of these photons which traces the path of the photon and checks the closest triangle which intersects with this photon. Once it intersects with a triangle russian roulette determines the probability whether it would get reflected or absorbed. This depends on the darkness of color since lighter colors tend to absorb less and reflect more. A random number is generated and photon is reflected is the random number turns out to be less than the russian roulette probability. The direction for reflection of a transparent surface is again determined by equation 3-5, for diffusive surface a random direction is chosen. when the ray hits the transparent surface, a random number is generated according to the reflectance at the certain interface to decide whether this ray is reflected or refracted. If photon gets absorbed (0,0,0) is returned as the color otherwise reflection occurs and photon moves in the new reflected direction but now with a reduced power. All such photons that bounce off from a specular surface on to a diffusive surfaces are then stored in the photon map (along with their power) using kd trees data structure [2]. Other photons are not stored because caustic effect is only formed from the photons that encounter a specular surface first and then go on to hit a diffusive surface.

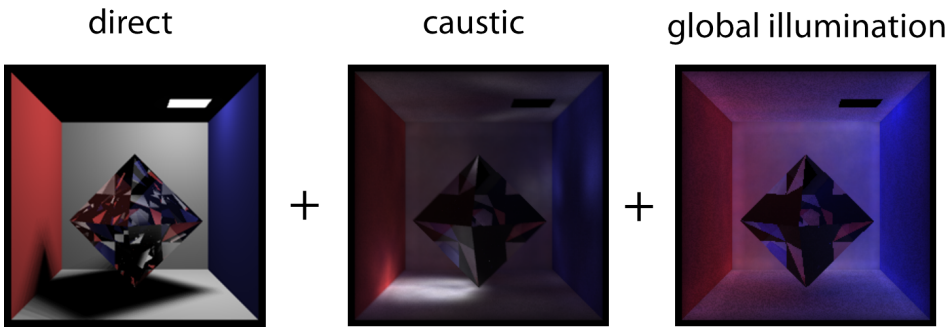
These recursive bouncing off from geometry ends when the power of photon becomes 0 or when the number of reflections have exceeded 10 similar to the ray tracing as explained earlier. We now have a photon map and we need to do modified ray tracing on top of it. These photon maps are stored in our implementation and can be reused to render different views of the same scene. In the second pass, we make rays originating at the camera and passing through each pixel similar to explained previously. Again intersections are checked and to speed up the process, we encapsulate the whole diamond inside an imaginary sphere and we first check the intersection of the rays with this sphere and ignore the rays which don't intersect it at all. The ray tracing here is exactly the same as the ray tracing in the previous section, except for the evaluation of the color of the diffusive surface. Once rays intersect with any triangle irrespective of their type, it tries to use the photons stored in that region in the kd tree in order to compute its color. We choose the nearest 500 photons to compute the color to average out the effect of noise. The pixel color is determined by the photon density in the region surrounding it. All other details are exactly the same as simple ray tracing just the way how color is computed is now different.

For caustic effects we form an imaginary sphere around the diamond and check it a photon passes through the sphere only then we store it. This is meant to ensure that caustic photon map stores only the photons that hit the diamond first and then a diffusive surface. This only gives use the bright shadow part. For rest of the lighting we use a global photon map, this stores each and every photon that either reflects off a diffusive surface or gets absorbed after the first bounce off the diffusive surface. As a result, every point a photon ever reached is now bright.

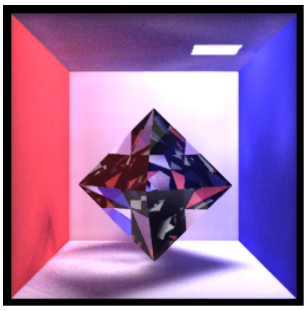
Figure 6 shows the result. We still use the basic ray tracing explained before to render the direct illumination shown as the first figure in Fig 6(a). This is the same as Fig 4(b). Then we use caustic photon map to render the second graph. 100,000 photons are stored and 500 photons are used for evaluating the brightness. One can observe the bright spot under the prism. The global illumination is rendered by using 100,000 global photons. Again 500 photons are used for evaluation. The ceiling has the reflected color from the blue and red walls. By adding these three graphs we got the realistic rendering of Fig 6(b).

3. CHALLENGES

One of the first challenges was to get a complex model of a diamond with high details. We started off by making an 8 facet prism manually. For a complex high detail diamond model we used the data set from The Perfect Diamond Project[1]. However, they were using a very specific format so we wrote



(a)



(b)

Figure 6: Apply photon mapping for caustic and global illumination. (a) rendering of the direct illumination, caustic and indirect global illumination. (b) the summation of the three figures in (a)

a python parser that took in their format which was similar to obj and wrote out the diamond model in our asc format.

Such a high level of detail brought challenges of its own. Since there were far too many faces of the diamond, at times light took 50+ reflections to get out from the diamond. Not compromising on the quality of image meant we had to put up with extremely long rendering times which were in order of hours. Initial part was implemented in matlab which took huge amount of time to render so we ported all our code to c++ in order to get a better rendering speed.

This project also required quite a bit of deep understanding of phyisics concepts. First we had to thoroughly understand whats going on under the hood in the photon mapping algorithm in order to get it implemented correctly.

4. CONCLUSION

In this paper, we laid out an overview of how we built a fully featured ray tracer that could render beautiful realistic diamonds. In our 5 step implementation, we described what ascpect each intermediate result lacked realism and how we fixed it. Our ray tracer has the capability of handling transparent objects, form soft shadows, caustic effects, global illumincation, dispersion and anti aliasing. Rendering our model from multiple angles enabled us to form an animation depicting a realistic rotating diamond .

5. REFERENCES

[1] Christensen *Ray Tracing For movies Cards* 2006: IEEE symposium

[2] Jenson, Henrik *Global illumination for Photon mapping 21.479.* Proceedings of the Eurographics Workshop on Rendering Techniques 1996.

[3] Josh Wiseman *The Perfect Diamond* 2005: Stanford