Implementation of a Bacic Monte Carlo Path Tracer

Mingyi Chen, Haoyang Liu

1. Introduction

Rendering is an important topic in computer graphics. It is the process of generating images from scene descriptions. Producing photorealistic renders requires the combination of application of computer science, geometric optics in physics and probability in mathematics; Monte Carlo path tracing is a widely adopted approach for producing physically correct images. Our implementation is a unidirectional path tracer. It currently supports effects including global illumination, glossy reflection and reflective caustics.

2. Background

We implemented a ray tracer in homework 3 of the Advanced Computer Graphics course. It is an improvement over the Whitted-style ray tracer, but it is not physically correct. For example, it does not consider light bounces between diffuse objects, resulting in missing global illumination for Lambert-like surfaces. On the other hand, the radiosity method is specially for such an effect, but it is quite expensive; combined with the requirement to tessellate the geometry along shadow boundaries, its use in practice is limited. Another major limitation of this method is that it cannot render specular or glossy reflections. Monte Carlo path tracing is a unified approach that can produce various realistic effects without the need for special handling as in many other methods.

3. Theory

3.1 Radiometry

In homework 3, the practice of treating light as electromagnetic radiations help quantify the measurements in the scene we are rendering. In this project we dive deeper for more accurate concepts. Radiometry defines the measurements for illumination and can help us better understand light transport and interaction with surfaces. Physically accurate renderers eventually come down to evaluating radiometric quantities. To get a more precise understanding of those quantities, we briefly go over some important concepts below.

Radiant flux is a measure of energy per unit time. It has the same unit as power.

Irradiance is the radiant flux per unit projected area.

Radiance is the irradiance per unit solid angle.

As we can see in the definition, irradiance is not a directional quantity. For Lambertian surfaces, where the appearance of objects is completely view-independent, irradiance can be precomputed.

Radiance can be considered to represent the power of one ray of light. It is a directional quantity. In a path tracer, when we trace a ray, we are trying to evaluate a radiance value from a specific point to the point of observation.

3.2 Monte Carlo integration

The rendering equation is a crucial theory behind physically based renderers. It provides a formula for the outgoing radiance from a point in a specific direction. A paper about the

rendering equation discussed the details of the formula. [1] A key for any renderer that implements that equation is choosing an integration method. Due to the potential complexity of the input scene, evaluating such integrals analytically is not possible in the general case.

The Monte Carlo estimator is

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

.[2] This estimator is unbiased, which means we can always get closer to the correct answer by taking more samples. It provides a general way for sampling the values from our integral to get an estimate of the result. It is powerful because it can be used to calculate the integral of very complex functions. The only requirement is the ability to evaluate the function at various sample points.

4. Implementation

4.1 Change of basis for domain of integration

This section describes a correction to the direct illumination logic in homework 3. The only kind of light source our renderer supports is the area light. The code from homework implemented jittered sampling for area lights. In the homework code, we treated each sample value from the light with equal weights, as if the sample rays projected uniformly onto the hemisphere. However, that is not correct. To utilize this sampler, we need to convert the domain of integration from the hemisphere to the area of the light. [3]

$$E(\mathrm{p}) = \int_{A'} L_o(\mathrm{p}', \omega')\, V(\mathrm{p}, \mathrm{p}')\, \frac{\cos\theta \cos\theta'}{|\mathrm{p} - \mathrm{p}'|^2}\, \mathrm{d}A'$$

4.2 Hemisphere uniform sampling

The next step is to implement a sampler for indirect illumination. For deciding the uniform hemisphere sampling, we initially started with sphere coordinates, as it is the most intuitive when it comes to hemispheres. For obtaining the hemisphere, we uses random θ and φ for the coordinate, then translate it to cartesian coordinate system.

Then the biggest problem is to transform it to the normal of the surface. The coordinate is generated with the normal at (0,0,1), so it has to be transformed. We used a transform matrix [4] for the purpose, But it has some flaws when it comes to certain normals, or to be precise, the vector pointing up. Since it involves dividing x and y, it will result in a div 0 error. Fortunately it is no big error and it can be bypassed by an extra if statement.

$$sqrt = \sqrt{x^2 + y^2}$$

The denominator part in the matrix

$$\begin{bmatrix} \frac{y}{sqrt} & \frac{-x}{sqrt} & 0 & 0 \\ \frac{x \cdot z}{sqrt} & \frac{y \cdot z}{sqrt} & -sqrt & 0 \\ x & y & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the matrix to transform the vector to normal (x,y,z).

Else than the uniform random sampling, we also experimented on sampling with an extra weight on a direction of the hemisphere to simulate glossy effects. The base idea is to first decide the chance of falling in the direction, for the part that is not in the direction, we just do a normal hemisphere sampling. For the part that is in the direction, we do an extra cone sampling to generate rays around the direction. The cone sampling is just a modified sphere sampling, with an extra limitation on the θ value, making it generating rays around the top of the sphere. In the algorithm, the normal of the sphere is not the parameter to pass in, but the desired direction for the ray is in place of the normal.

4.3 Combining direct and indirect illumination sampling

The above sections (4.1 and 4.2) complete the prerequisites for an importance sampling scheme. In a scene, we usually expect that most of the incoming radiance contributions at a point is from direct lighting. The user can specify the number of direct light samples to take from each point, while the number of indirect illumination samples is always one, which is the standard case for path tracing. The result is shown below in Figure 1.

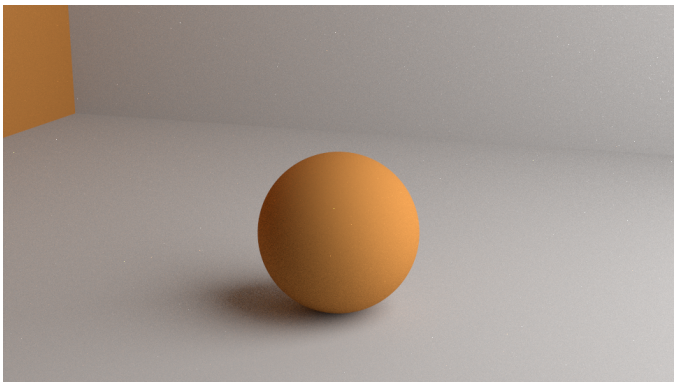

Figure 1.

Notice the shadow is not completely black. This is because of indirect illumination. No ambient term is included in our renderer. There is also color bleeding from the wall and the sphere to the floor.

4.4 Extending BRDF: glossy

Glossy effects depend on the angle of observation. The previous BRDF simply distributed the differential irradiance at a point uniformly over the hemisphere, producing a purely diffuse look. The addition we made to the BRDF is to calculate the angle between the reflection of the incoming ray and the outgoing ray. The closer the angle, the more light should be distributed in that particular outgoing direction. The distribution is determined with the power of the cosine function. The next step is to normalize the distribution so that energy is conserved. The result is shown in Figure 2.

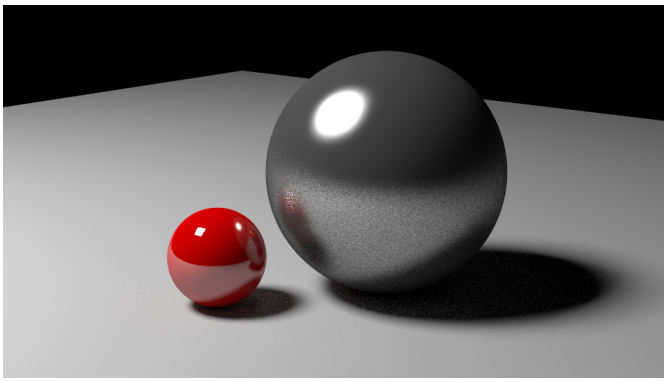

Figure 2.

The red sphere is perfectly smooth, but the gray sphere has some level of roughness.

4.5 Reflective caustics

Specular reflection is computed separately. When using a uniform hemisphere sampler, we cannot aim for a particular incoming direction. But because of the uniform nature of the algorithm, we can get reflective caustic effects without additional logic. Previously we achieved color bleeding. Caustics can be seen as the first diffuse object changed to a reflective one. The result is shown in Figure 3. However, because our path tracer is unidirectional, caustic effects are relatively costly to render.

Figure 3

5. Defects and Limitations

Noise is one of the common problems in Monte Caro path tracers. Sometimes there are noisy sections that are difficult to eliminate with increasing samples. We must determine if that is a systematic error, or a sampling problem. Figure 4 shows a sampling problem. It also demonstrates how the estimator may suffer from high variance when the sampling scheme does not fit the underlying function. At points near a large area light, each point receives uneven distributions of radiance coming from each sample point on the surface. The uniform sampler we are using is vulnerable to uneven distributions, resulting in an unusual amount of noise for points near the light.

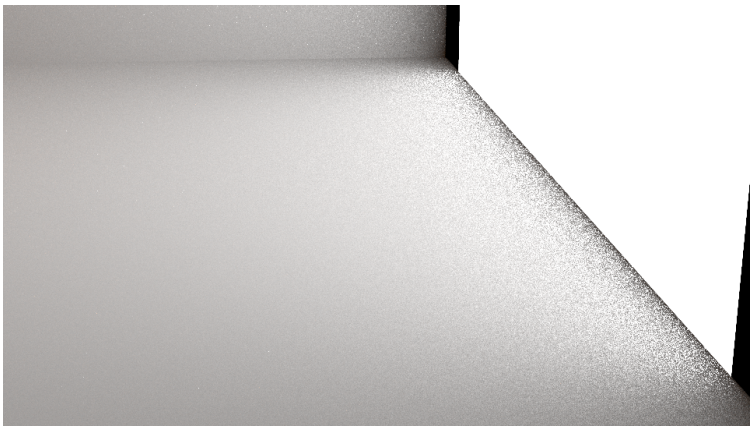

Figure 4. Points close to a large area light are noisy.

One limitation of our renderer is that it does not support the Fresnel effect. Fresnel effects are very common in the real world. For some reflective materials, the flections are clearer when viewing at an angle further from the normal.

6. Conclusions and Future work

We introduced our implementation of a basic Monte Carlo path tracer. It can render basic effects required by modern physically based renderers.

Path tracers can be complex and there are many possibilities for improvements. As future work, we can implement Fresnel effects for reflective objects. In addition, sampling strategy is an important factor in the performance of any Monte Carlo path tracer. We wish to implement a sampling strategy that adapts to the radiance distribution, which can effectively reduce the variance of the result.

Reference

[1] Kajiya, James T. “The Rendering Equation.” *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, 31 Aug. 1986, pp. 143–150., https://doi.org/10.1145/15886.15902.

[2] Pharr, Matt, et al. “The Monte Carlo Estimator.” *Physically Based Rendering: From Theory to Implementation*, Morgan Kaufmann, Amsterdam, 2017, pp. 751–752.

[3] Crane, Keenan. “Monte Carlo Rendering”. http://15462.courses.cs.cmu.edu/fall2020/lecture/montecarloraytracing/slide_023

[4] amd. “Getting a Transformation Matrix from a Normal Vector.” *Mathematics Stack Exchange*, 6 Oct. 2016, https://math.stackexchange.com/questions/1956699/getting-a-transformation-matrix-from-a-normal-vector.