1 Project summary *(from proposal)*

I plan to explore "Monte carlo ray-tracing." MC ray-tracing comprises a number of ray-tracing methods all of which involve using stochastic approximations to the rendering equation, the equation giving the radiance at a point as a function of a direction away from that point (and toward, say, a camera/eye). I plan to at least implement a monte carlo ray-tracer as described by by Peter Shirley "Direct Lighting via Monte Carlo Integration," chapter 3 of Siggraph 2003 Course 44. The main advantage over a basic whitted ray-tracer is in global diffuse lighting. However, I also plan to implement photon mapping. What is guiding me is obtaining realistic caustic effects. More details are in the "Milestones" section below.

2 Milestones *from proposal*

1. *MC ray-tracer.* My first set of tasks will be to implement the basic MC ray tracer, without regard to efficiency. I have made progress in this direction (see below) and over the next week plan on implementing the ray tracer for planar light sources. I will try to replicate the results that Peter Shirley presents, using different planar light sources and different numbers of samples. I should then attempt to handle more complicated light sources. At least, perhaps, spherical light sources as in Changyaw Wang, "Physically correct direct lighting..." in Graphics Gems III. Shirley in the above-cited chapter provides a method for handling light sources with varying intensity across their surfaces, with constant intensity as a sub-case, and I may attempt this generalization.

2. *Acceleration methods.* Here I will attempt to accelerate the ray tracer by the choice of the distribution function used in the MC integration. In particular, I will examine stratification, as discussed in chapter 6 of the Siggraph notes and general MC texts, e.g., G. Fishman, "Monte Carlo: concepts, algorithms, and applications" (1996). By varying the parameters of the integration I will be able to assess realism/speed trade-offs.

3. *Photon mapping.* First I will implement path tracing, in order to obtain diffuse interreflections. Next I will work on building global and caustic photon maps. This involves sampling a point on each area light, then sampling a direction, then forward tracing a ray (the "photon") from that point and in that direction through the scene. At points of intersection with the scene, the power of the photon is stored, thereby building a photon map (typically a kd-tree).

3 Report

I implemented a basic MC ray-tracer with various shapes as constant intensity light sources (milestone (a), omitting light sources of varying surface intensity) and the planned acceleration methods in milestone (b). Unable to produce convincing caustic effects using these methods, I implemented photon mapping (milestone (c) to get better caustic effects. Below I describe the project, give some background, and describe my process. The text is a little long-winded so I added many pictures with captions and these should be enough to convey the substance of this report.

As mentioned, a stochastic approximation to the rendering equation is used, giving the radiance at a point as a function of a direction away from that point (and toward, say, a camera/eye):

$$L_o(x,\omega,\lambda,t) = L_e(x,\omega,\lambda,t) + \int_\Omega f_r(x,\omega',\omega,\lambda,t) L_i(x,\omega',\lambda,t)(-\omega' \cdot n) d\omega' \tag{1}$$

Here $L_o(x,\omega,\lambda,t)$ is the radiance in the direction ω at wavelength λ at time t; L_e is the light emitted at the point; f_r is the bidirectional reflectance distribution function, providing the likelihood of reflection in outgoing direction ω given an incident direction ω'; L_i provides the radiance arriving from direction ω';

and the inner product $(-\omega', n)$ between the incident direction and the normal at the point being rendered represents the attenuation of light as the incident angle departs from the normal. (The subscripts and notation may be confusing, but I am following convention. L_o and L_i are the same function, evaluating radiance, and the subscript really refers to the argument–whether the exitant or incident radiance is under consideration. The equation is therefore recursive. L_e, the emission at the point, is different.) The integral is taken over all incident directions ω'. This set of directions may be identified with the hemisphere on the tangent plane with major axis the normal. As the equation states that the exitant light equals the emitted light plus the reflected light, it is apparently an instance of conservation of energy. For my purposes, I will not consider the time parameter (scenes will be static). The equation is usually attributed to the 1986 paper by James Kajiya. A fuller derivation is provided in Pharr & Humphreys 2004.

I used MC integration in two separate parts of my ray-tracer. One was path-tracing, which is Kajiya's original approach to global illumination. The other was in sampling area lights.

3.1 Path tracing

Since the rendering equation has radiance terms L on each side, it invites a recursive solution. Path tracing involves tracing a path from the camera back through the scene a certain number of steps, where each step is an intersection with an object in the scene, and calculating the illumination along each step. Light is assumed here to travel in straight lines and leave surfaces according to the BRDF for the surface. Therefore, this path simulates in reverse a ray of light traversing a scene from a light source to a camera/eye.

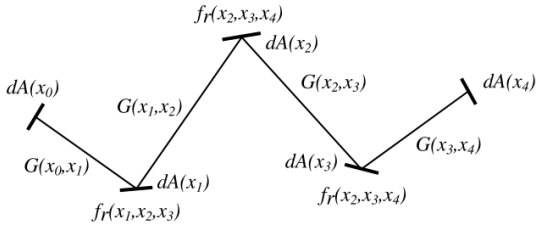

Figure 7.3: A path from point $\vec{x}_1$ to $\vec{x}_5$.

Figure 1: Figure from Pat Hanrahan's contribution to Siggraph 44.

In more detail: first, to provide the initial energy in the system, there must be some emitters of light. These are evaluated by the L_e term in the rendering equation. Second, a path must terminate if it intersects with no object in the scene. However, to bound the recursion in scenes such as an enclosed box, where there will always be an intersection. I used a random termination time sometimes termed "Russian Roulette." Given a value to estimate by recursion, such as the radiance L at a point, the recursion is terminated with a certain fixed probability α. Upon termination 0 is returned. If the recursion is not terminated, the recursion proceeds but the estimate is scaled by $1/(1-\alpha)$ and returned (in this example, $L/(1-\alpha)$ would be returned). Thus the expected value of the estimate is $(1-\alpha) \cdot L/(1-\alpha) = L$, so the estimate is unbiased. (RR may affect convergence speed, however.) Some authors suggest putting a hard bound (which would then bias the estimate). I used the basic form of RR. RR is described in Pharr & Humphreys 2004 as well as Siggraph 44.

The scene in Fig.2 was made with my path-tracer. I augmented a basic (local/Whitted) ray-tracer I found on the Internet, but may adapt our code from assignment #4 if time permits. The improvements over the basic ray-tracer is in the spread of diffuse light around the scene. The path tracer will pick up this global effect. A local ray-tracer can and does recurse, but only in the directions of specular and refractive light. Diffuse light comes in from every direction. Since it is not practical to evaluate for every direction, we must sample from the set of available directions. Doing so is the substance of MC ray-tracing. In practical terms, this involved computing the approximation to the rendering equation, i.e., $L_o(\omega_o) = L_e + brdf(\omega_i, \omega_o) \cdot L(\omega_i) \cdot pdf(\omega_i)$, where the Lambertian BRDF is simply Reflectance(point being modeled)/π for a diffuse surface (the π is required to ensure conservation of energy, as discussed in section 9.1.1 of Pharr & Humphreys 2004).

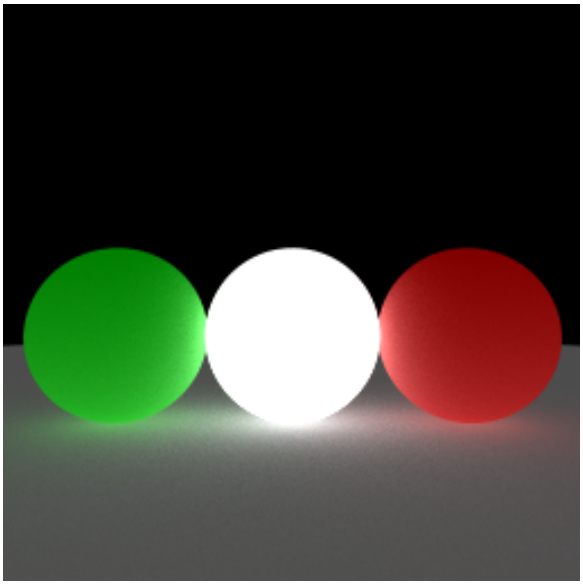

Figure 2: My simulation of a scene found in the literature. The diffuse light is an improvement although the middle bulb may be too white.

Figure 3: "Cornell box." All lighting is diffuse interreflection (besides startup emissive energy). The spheres appear somewhat flat, maybe because there is no specular reflection. (250 samples/pixel.)

Fig.4 shows two other examples using boxes. In all these scenes there aren't the idealized light sources, such as point lights, only area lights, i.e., shapes with nonzero emission. I also implemented instancing to allow transformations of the light sources (and other shapes).

Following a suggestion of Shirley, I do a simple test of correctness of a basic function of the path-tracer. The general form of the recursion is $L_o = L_e + RL_i$, where the L_i term is the point of recursion. Holding L_e and R fixed in the scene, the first evaluation is then L_e, the second is $Le + RL_e$, the third is $L_e + RL_e + R^2L_e, \ldots$, the geometric series with ratio R, converging to $L_e/(1 - R)$. Placing the camera in a box to ensure an intersection at every vertex, and setting the box properties to .5 for the emission L_e and (.5,.5,.5) for the diffuse reflection constant R, I thus use the path-tracer to approximate the series converging to .5/(1-.5) =

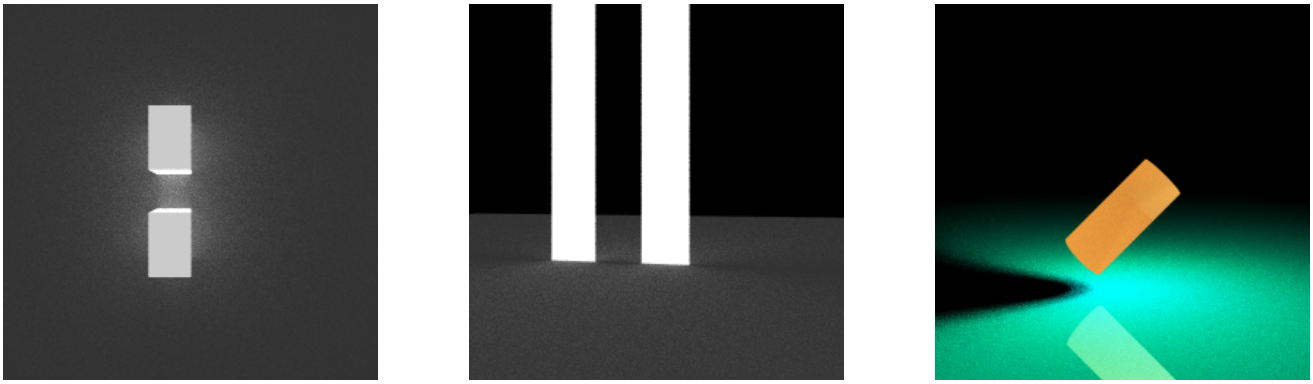

Figure 4: Three views of a scene with only area lights, i.e., shapes with nonzero emission. The last also demonstrates the instancing implementation, with an emissive cylinder rotated. Only the sides of the cylinder are emissive, creating hyperbolic shadow.

1 (for each of the R, G, and B channels):

```
1 path segments: 0.5,0.5,0.5
2 path segments: 0.75,0.75,0.75
3 path segments: 0.875,0.875,0.875
4 path segments: 0.9375,0.9375,0.9375
5 path segments: 0.96875,0.96875,0.96875
6 path segments: 0.984375,0.984375,0.984375
7 path segments: 0.992188,0.992188,0.992188
8 path segments: 0.996094,0.996094,0.996094
9 path segments: 0.998047,0.998047,0.998047
10 path segments: 0.999023,0.999023,0.999023
11 path segments: 0.999512,0.999512,0.999512
12 path segments: 0.999756,0.999756,0.999756
```

3.2 Using MC to approximate direct lighting (soft shadows/shadow rays)

Another use of MC integration is sampling light sources to compute direct lighting. Ad hoc techniques are needed to accelerate convergence. Consider a scene with a large emissive sphere and a smaller non-emissive sphere (fig.5). A naive path-tracer will choose a random direction anywhere on the hemisphere about the normal at the point being modeled, which may rarely coincide with the direction of a light source. In such a situation the effect of the emitter will be slow to accumulate, as in fig.5, where the effect on the non-emissive sphere and the ground is only slight after 10 minutes on my machine (about 2Ghz, 2 Gb RAM). The illuminated sphere and ground are faint and may require adjusting one's monitor or viewing angle to make out. (By contrast, if the hemisphere on the tangent plane is trained on an emitter, as in the points of tangency of the three spheres in fig.2, the incident illumination may be nearly total/white.)

Therefore, it is common to sample emissive objects directly to speed up the convergence of path-tracing. I read about this in Pharr & Humphreys 2004 and also Shirley's contribution to Siggraph 44. Fig.6 has my imitations of trials run by Shirley to test the soft shadow effects created by his direct lighting MC simulator in Siggraph 44.

Sampling light sources proved difficult, even though the only shapes I sampled were spheres and boxes (other than 2-D polygons). The horizontal shadow beneath the square in the lower right of fig.6 is a relic of my initial attempt at sampling boxes. As the above equations show, the pdf of the sample is required for an MC estimate. When sampling uniformly on a sphere the pdf is straightforward to calculate: assuming the entire sphere is available for sampling, it is proportional to the reciprocal of the surface area. (Uniform sampling

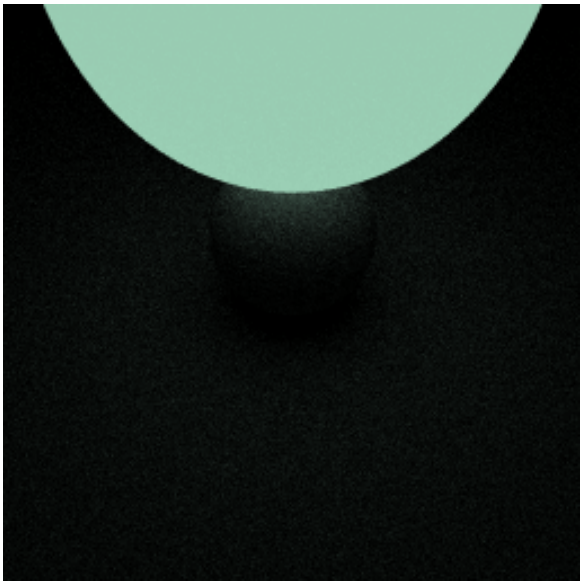

Figure 5: A naive path-tracer, issuing a ray in a random direction from the point being modeled, takes forever to render illumination–the smaller sphere is hardly illuminated by the large emissive sphere.

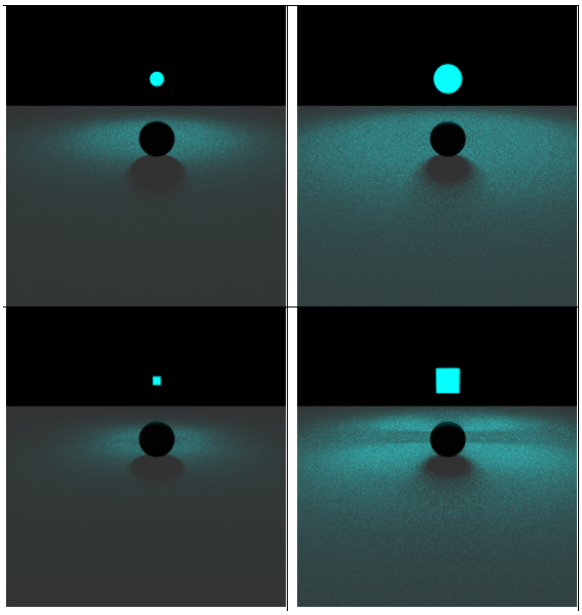

Figure 6: Accelerating the rendering of illumination by directly sampling light sources.

a sphere may itself be inefficient, since the far side of the sphere from the point being rendered will make no contribution, but sampling only in the cone subtended by the sphere is a further optimization.) This is because the sphere looks the same from any angle. The pdf when sampling from a cube is more complicated, and I didn't find discussion in the references I had available. The strange artifact in the lower right of fig.6, the horizontal shadow below the sphere, is what happens when one simply uses the surface area of the entire parallelepiped, pdf $= 1/(2 \cdot (\text{area of first face}) + 2 \cdot (\text{area of second face}) + 2 \cdot (\text{area of third face}))$. My next attempt was to only use the area of the sides visible to the point being modeled. There are at most 3 of these (unless the point is inside the cube). This pdf did not lead to the correct result either (fig.7), but from this image my error becomes clearer. The discontinuities occur where the there is a jump in the number of sides visible from 1 to 2. As one moves continuously from a point from which only one side is visible, to a

point from which two sides are visible, the visibility of the newly visible side should increase continuously. Therefore, to use the surface area of one side when one is visible and suddenly the area of two sides when one is visible and a second is just slightly visible leads to inaccurate results. I used the cosine of the angle between the ray between the point being modeled and a given side (I actually averaged over the four points of the cube's face), and the normal to that side, to scale the area of that side and thus obtain more realistic results. Conceptually, I am sampling on the projection of the box on the hemisphere at the point being modeled.

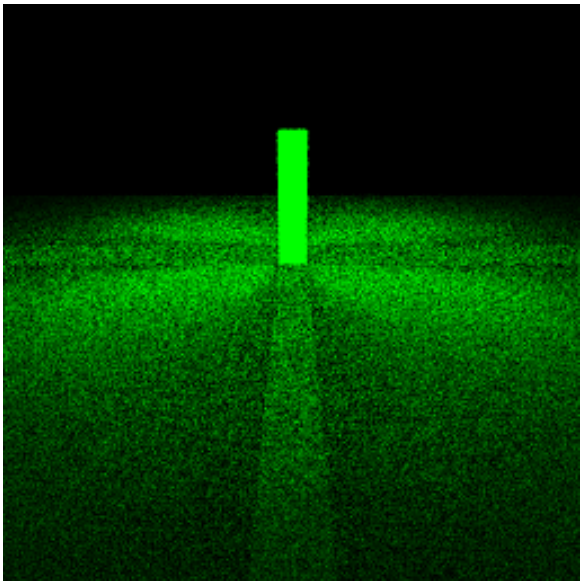

Figure 7: A bad attempt at sampling from the surface of a parallelepiped, sampling from the 3 visible faces.

One still notices that there is less light directly underneath the cube and along the strips parallel to the cube's side; but this may be the correct effect, since less light is visible at those points. I am actually unsure about how light actually behaves in this situation, so I can't really judge the simulation's effectiveness. At least, the transitions are now continuous. (Fig.8.)

3.3 Specularity/transmission

Stochastic methods are less important for modeling reflection/refraction. In the idealized model, the direction of specular/refractive reflection is determined (the reflection about the normal in the plane determined by the incident ray and the normal for specular reflection, and the direction given by Snell's law for refraction).

Reflection/refraction can be cast in the MC framework, as Pharr does, and as I do as a check on the framework. We simply use a delta/impulse distribution with the mass concentrated in the direction of perfect reflection/refraction. The delta distribution shows up in the BRDF in the numerator of the MC estimator and in the pdf in the denominator, so that the algorithm is effectively the same as the non-stochastic approach discussed in lecture. (Fig.9.)

The MC framework does provide one opportunity to improve on this original reflection/refraction code somewhat, by manipulating the BRDF. I used the Fresnel approximation of the amount of energy reflected/absorbed for dielectics (transmissive) and conductors (non-transmissive) as the BRDF. This allows for manipulation of the reflection/transmission behavior by setting Fresnel constants associated with the dielectic or conductor. (Fig.10.) These are still hard-coded in my program, however, although it would not be too difficult to extend it to read this in from scene files; this remains *to do*.

There are specular BRDF's that do take advantage of MC tracing. These are physically based but do not reflect light with the same idealized single-direction model as perfect specular reflection. Two in particular

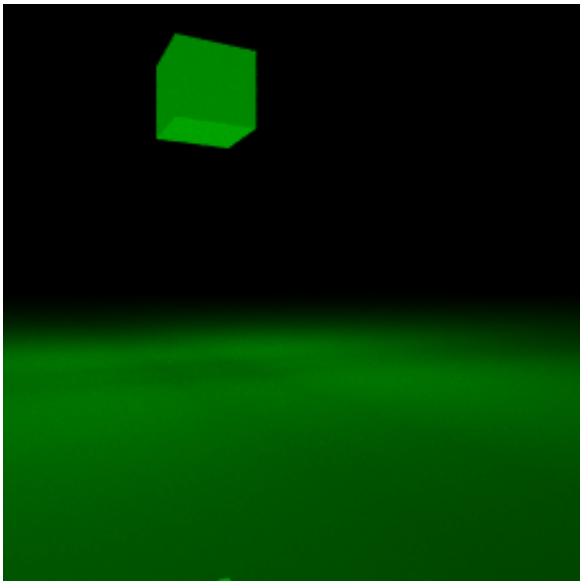

Figure 8: An improved attempt at sampling from the surface of a parallelepiped, sampling on its projection on the hemisphere about the normal at the point being modeled. Is the shadow under the cube the correct physical behavior?

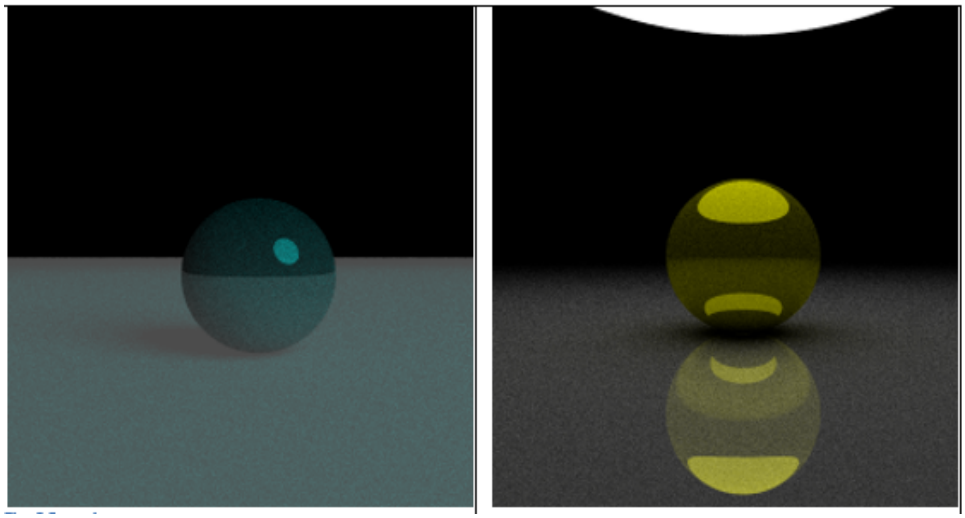

Figure 9: Using a dirac delta to treat reflection/refraction in a stochastic framework. Doing so serves to check whether the stochastic ray-tracer properly generalizes the non-stochastic. Reflection is from an emissive sphere; there are no direction lights in this ray-tracer, so the specular effect doesn't soften at its edges, giving an almost cartoon-like effect.

would be straightforward to implement: the Blinn microfacet model and an anisotropic model described by Shirley. Implementation would only require supplying the code for the BRDF, for which formulas are provided in various sources, e.g., Shirley 2000. However, this remains *to do*.

3.4 Optimization

I used bounding box checks to accelerate intersection. Two other acceleration techniques I implemented are stratified sampling and importance sampling. Stratified sampling is probably the more straightforward. It

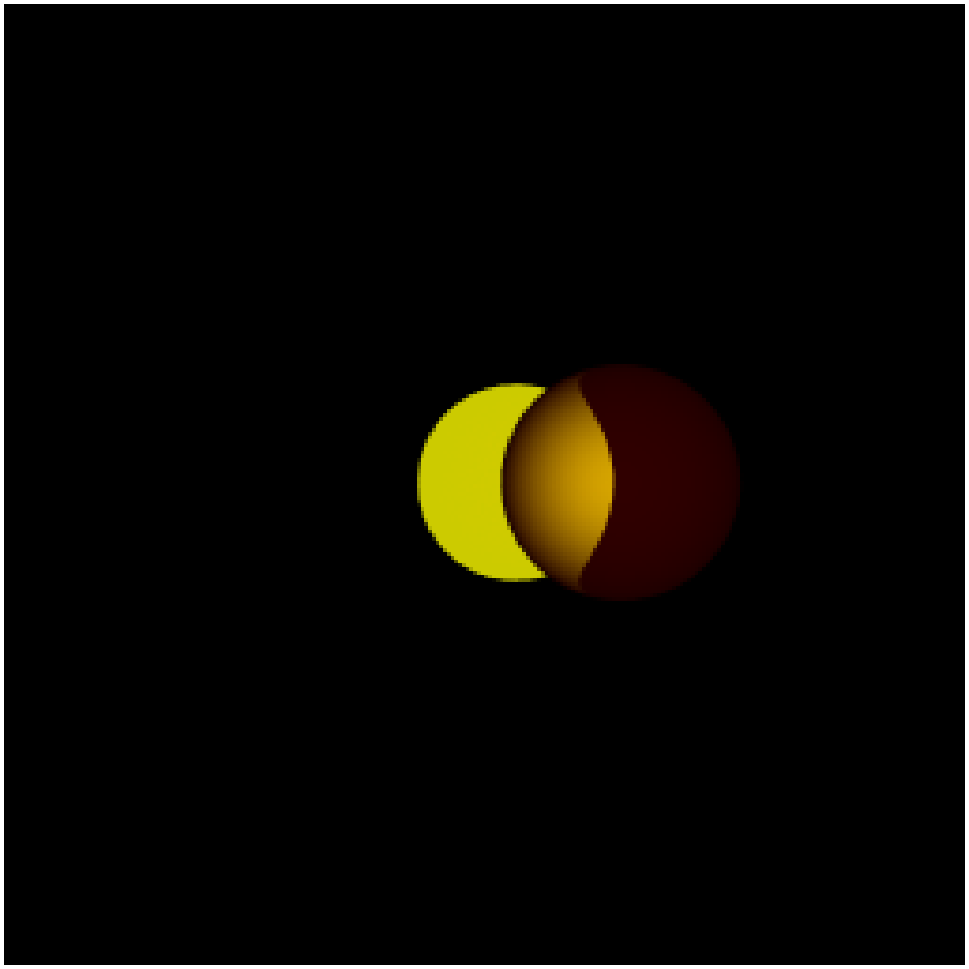

Figure 10: Using the Fresnel approximation of the amount of energy reflected/absorbed for dielectics (transmissive) and conductors (non-transmissive) as the BRDF.

involves partitioning the range of the random variable and restricting values to each of the partitions, to ensure that the values occur at least in each of the partitions. The simple manner in which I use this is in partitioning the area around a pixel into four quadrants, and issuing a ray randomly from within each quadrant, thereby ensuring the pixel sampling doesn't clump around the same area (e.g., just two of the quadrants). For example, in fig.11 I show unstratified and stratified sampling from a hypothetical pixel-area square. It so happens in this unstratified set that there appear to be fewer samples on the left (negative x direction); stratified sampling with a central vertical partition can be used to force a balance. Below the balance is enforced across the horizontal and vertical.

One can also imagine stratified sampling being used in distributed ray tracing in sampling the hemisphere at a point for diffuse interreflection, by partitioning the hemisphere into various regions. This application of stratified sampling would be somewhat more complicated, requiring adjustment to the pdf of the sample rays through the hemisphere accordingly, in order to use the MC estimate. Importance sampling involves a judicious choice of the distribution function used in Monte Carlo integration. The description of MC given in the first section makes no requirement on the pdf; convergence is guaranteed by the LLN. However, the speed of convergence may be greatly affected by the selection of a pdf that makes the input x more likely according as $f(x)$ is greater, i.e., the pdf is similar in shape to the integrand. In such a way, samples will make a larger contribution to the approximation, whereas integrand values of smaller magnitude and effect on the approximation will be less likely to be sampled.

Cosine-weighted sampling on the hemisphere is a common example. In calculating diffuse interreflection,

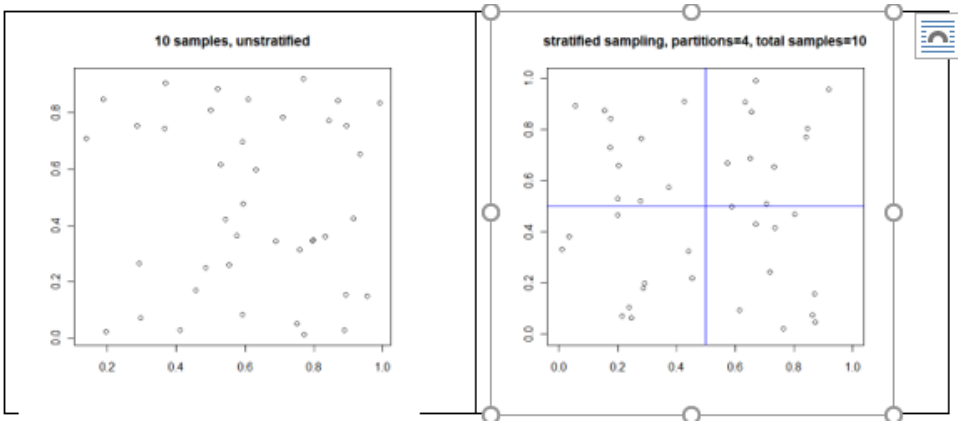

Figure 11: Unstratified vs. stratified sampling.

one may sample uniformly on the hemisphere about the point being rendered. However, samples near the "equator," i.e., farther from the normal, will contribute little, with no contribution at points orthogonal to the normal, due to the cosine term in the rendering equation. Accordingly, an application of importance sampling is to weight the sampling toward the normal or "north pole" of the hemisphere rather than using a pdf uniform on the hemisphere. A pdf proportional to the cosine of the angle between the random vector and the normal will weight samples toward the normal. Figures *** compares sampling using sample_hemisphere and sample_hemisphere_cosine ***, using the same number of samples. *TODO.* Other images are presented in figs.(12,13).

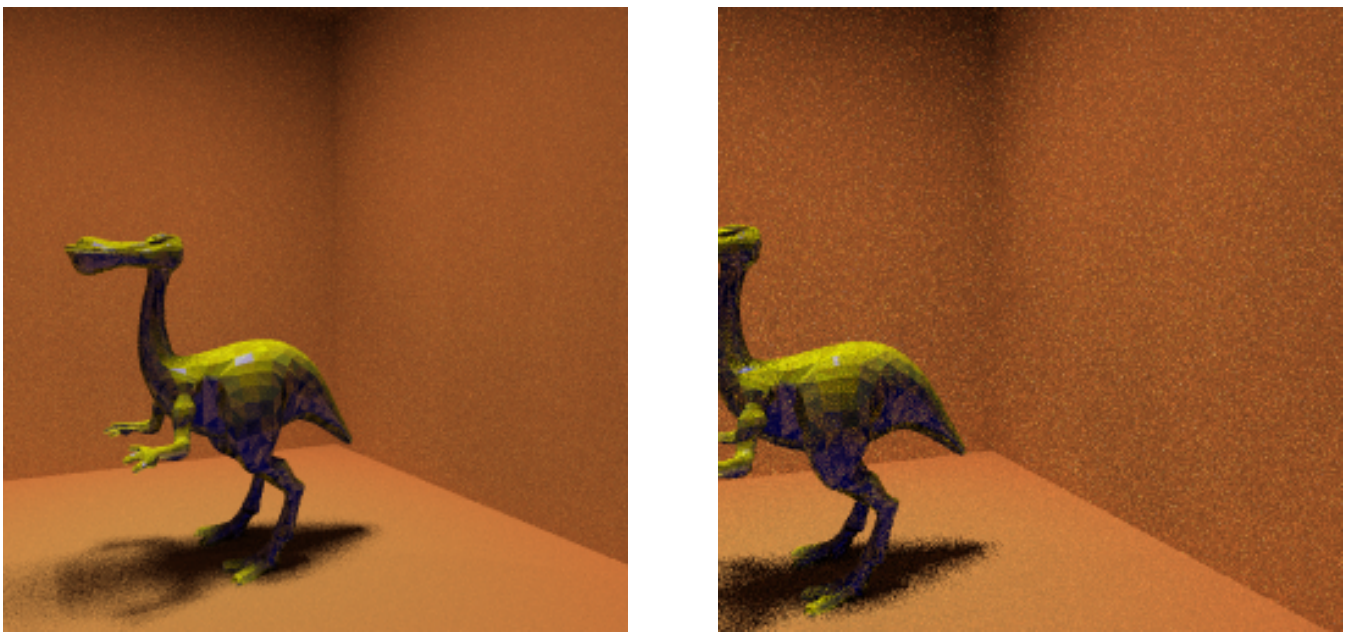

Figure 12: Reflective dinopet, with 50 samples/pixel (left) and 10 samples/pixel (right).

My first problem with the MC ray-tracer was too-heavy shadows as in the left scene of fig.13. The sphere is transmissive, yet very little light appears to make it through. The problem is that the direct lighting contribution is mostly nonexistent: using reverse ray tracing it isn't feasible to sample directly from the light source through a refractive medium, since if the index of refraction is nontrivial and the light source isn't huge, single rays issued from the shadow region will not strike the source. One way out is to use forward ray tracing (bidirectional path-tracing in this context). Another way is photon mapping, which issues photons in the forward direction (*TODO*; this is milestone (c)). But there is still the contribution to the irradiance made by diffuse interreflection, albeit is much smaller than the direct lighting contribution. Therefore, there

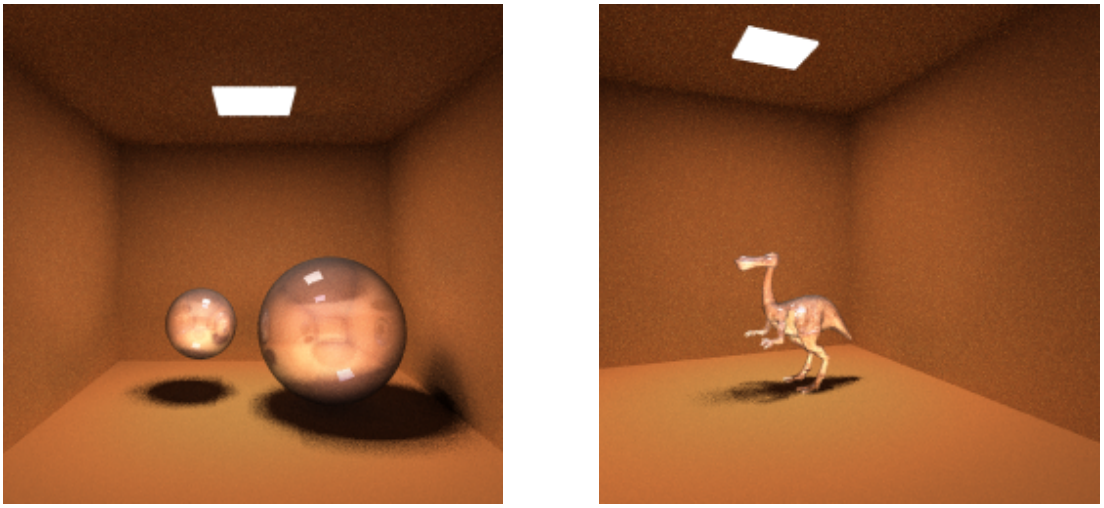

Figure 13: Transmissive dinopet (right) and a buggy attempt at caustic effects on the left. Shadows are probably too strong also. The diffuse interreflections arent effective below the spheres; the stochastic rays issuing from those points apparently arent making it through the transmissive spheres to pick up light.

is something problematic about the scene above. It should not appear as complete occlusion. The stochastic rays issuing from the area under the sphere should still boost the light in the region somewhat. This turned out to be a bug in how I was randomly sampling directions on the hemisphere about the normal. Actually, I am surprised the previous images worked so well (or that my ability to detect departures from the correct behavior is so poor), given this defect central to the diffuse interreflection code. Uniformly sampling a direction on a sphere is relatively straightforward, given uniformly distributed α, β, take $\theta = \alpha * 2\pi$, and $\phi = cos - 1(\sqrt{\beta})$, as discussed in Pharr, Wikipedia, etc. The difficulty I found was finding the basis vectors to use at the point of intersection, once one has obtained the direction, given by θ and ϕ, and needs to convert to rectangular coordinates. I did not encounter algorithms in the literature so I first tried

```
N = normal vector at intersection point
N2 = N
If N[0]!=0) N[0] = 0;
   Else if N[1]!=0 N[1] = 0;
      Else N[2] = 0;
U = CrossProduct(N,N2)
V = CrossProduct(N,U)
```

Except for some exceptional cases (e.g., the normal is $(1, 0, 0)$) that can be handled individually, this approach will give an orthogonal basis at the point of intersection, with N one of the basis vectors. This algorithm was an improvement over my previous buggy code.(Fig.14.)

The left scene in fig.15 was set up to check for caustic effects. The caustic shape is noticeable, but the power of the caustic is not appreciably greater than that of the other reflected light on the surface, whereas one expects a caustic to stand out. One also notices another problem: the yellow reflected light appears to have a certain motion toward the right in the image, as though the light source were at an angle. The source is parallel to the surface, so the reflected light should be symmetric. The problem is in my algorithm for finding a basis at the intersection point, which is biased in zeroing out (in order to obtain a direction different from the normal) the first component if it is nonzero, failing that the second, and failing that the third. I therefore tried choosing the direction to zero out randomly. This generated a correctly symmetric reflection, as in the right scene in fig.15. My understanding from Henrik van Janssen's text on photon mapping is that caustic effects are difficult to obtain using conventional ray tracing.

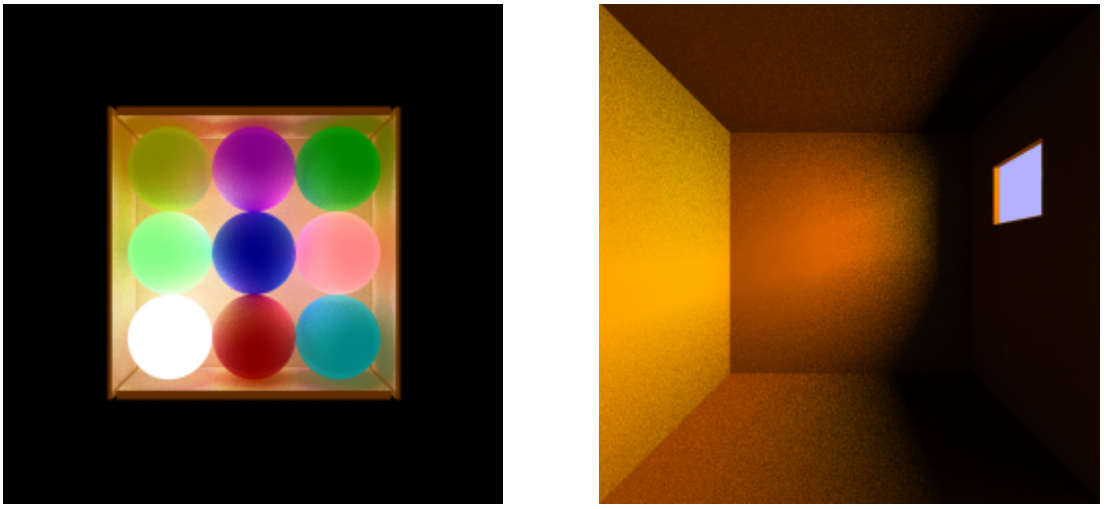

Figure 14: Redoing fig.3 with corrected diffuse interreflection code, and a new scene.

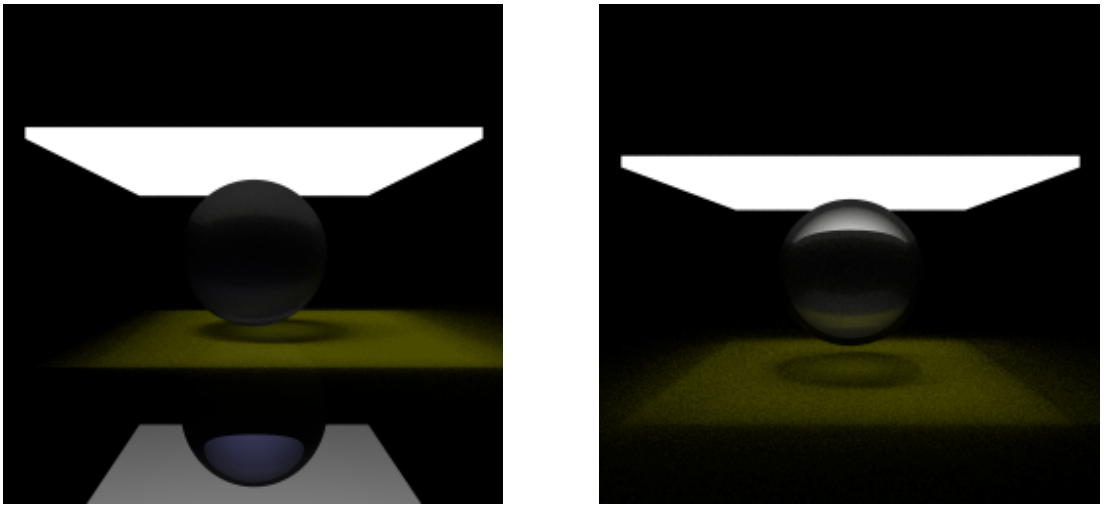

Figure 15: Attempting to obtain caustic effects. The asymmetry in left image arose from an inadvertently introduced bias in my sampling routine, corrected on the right. The caustic effect is present but muted. Hopefully the caustics will be improved using photon mapping in the next part of the project.

3.5 Photon Mapping

In order to implement photon mapping I completed Assignment #3 and the extra credit of CMU's 2005 Advanced Graphics course. The assignment page is at

`http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15864-s05/www/assignment3/`.

The skeleton code provided a Whitted-style ray-tracer and van Jensens implementation of the photon map data structure and irradiance estimator, taken from van Jensen 2001. Also included was a debugging interface for viewing ray and photon intersections with the scene. The goal was to get this Whitted tracer to implement path tracing and photon mapping.

3.5.1 Path tracing

The first part of the assignment was to implement path tracing in order to obtain diffuse interreflections. This was basically what I set out to do in this final project (see above), i.e., sampling path directions from the hemisphere at the point of ray-surface intersection, and sampling from area lights. Actually the CMU assignment only required sampling from square area lights, whereas I spend most of my time in the before the milestone figuring out how to sample uniformly from different shapes. The code is in `material::shade`. Soft shadows are probably the most salient effect of this feature. Fig.16 three scenes show a Cornell box using the supplied Whitted tracer, then with path-tracing with 1 sample/pixel edge, then with path-tracing with 5 samples/pixel edge (`cornellbox.ray`).

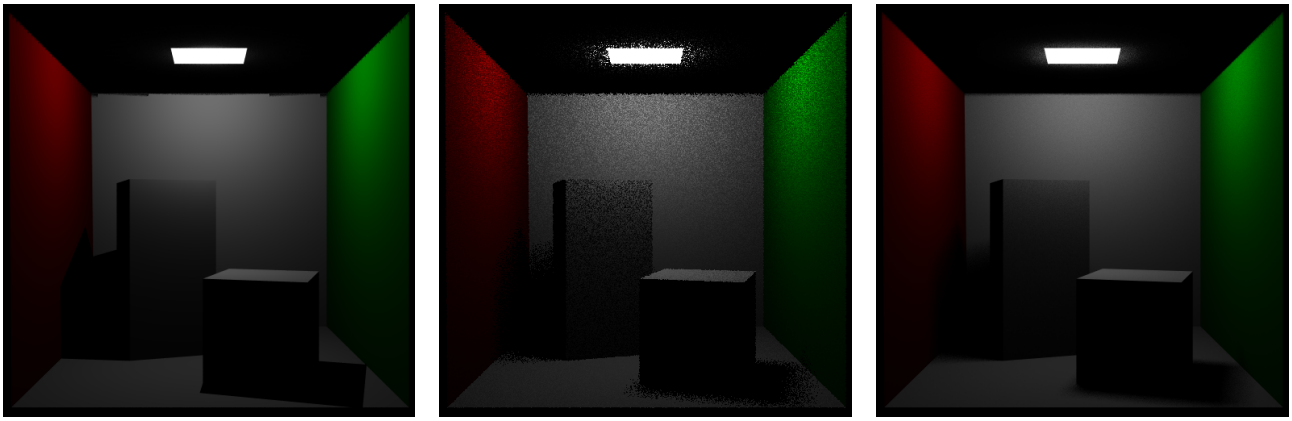

Figure 16: Adding path tracing to the CMU code. The left scene using the skeleton code, a Whitted ray-tracer, has hard shadows and aliasing at the top of the rear wall. In the right two, using path tracing with 1 and 5 samples/pixel, shadows are softer and aliasing is removed.

3.5.2 Building the global and caustic photon maps

The next step is unique to photon mapping. The first phase of photon mapping algorithm is building the photon maps. This involves sampling a point on each area light, then sampling a direction, then forward tracing a ray (the "photon") from that point and in that direction through the scene. At points of intersection with the scene, the power of the photon is stored, thereby building a photon map. (The photon map is a kd-tree; it was provided in the skeleton code I used.)

In particular, at each intersection, "Russian Roulette" stochastic termination (see above) is used to decide whether the photon will be reflected, refracted, diffusely scattered, or absorbed. The global photon map records the power of the photon at all intersections with diffuse surfaces, whereas the caustic photon map only records those intersections with diffuse surfaces that have been preceded by specular reflection or refraction. Abusing regular expression notation, the paths recorded in the global map are conventionally described as $L(D|S)*D$ and the paths in the caustic map as $LS+D$, where L denotes emission from the light, S denotes reflection/refraction, and D denotes intersection with a diffuse surface. The code is in `RayTracer::trace`.

Perhaps it is easier to describe the building of the photon maps by making use of the CMU tracer's supplied debugging facilities. Fig.17 shows debug views of the building of the global map.

The white rays represent stochastic photon paths and the white dots in the scene are photons stored in the photon map. The photons are stored using spatial coordinates; the photon map is decoupled from the geometry of the scene. The green segments are normals at several points of intersection. A caustic photon map built for the same Cornell box scene is empty (fig.18 left). The reason is that this scene contains no reflective or refractive surfaces, and caustic photon paths by definition intersect with such surfaces. When reflective boxes are used the situation is quite different (fig.18 right), and similarly when transmissive spheres are used (fig.19).

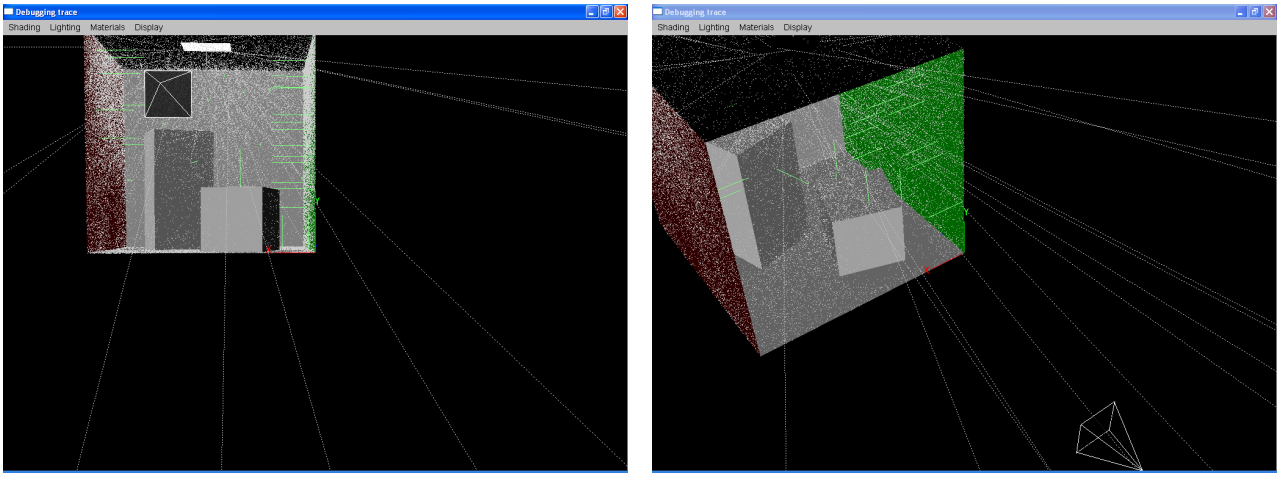

Figure 17: Debugging views of the global photon map.

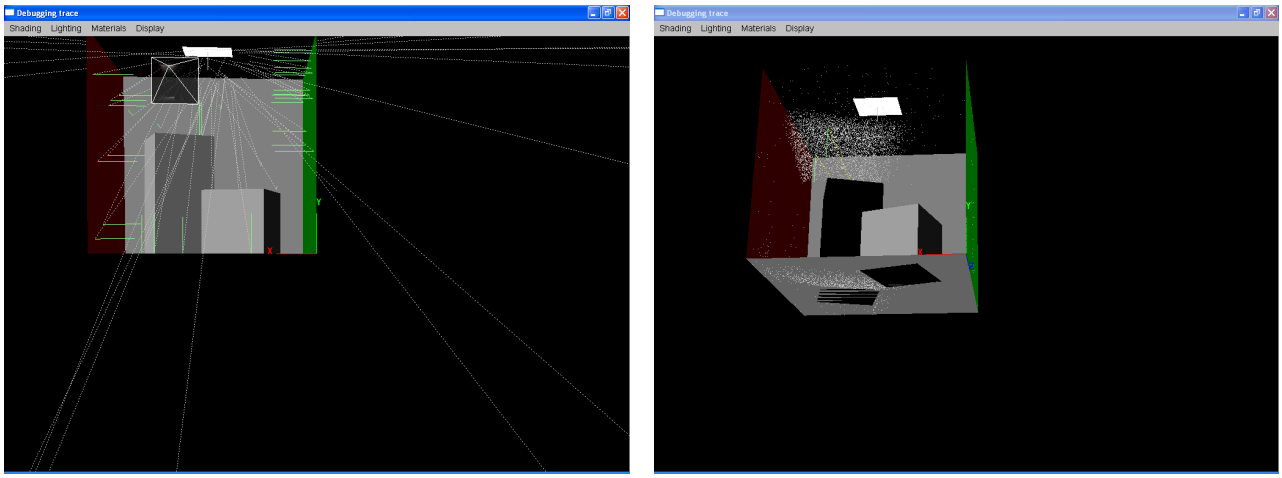

Figure 18: Debugging views of the caustic photon map with all surfaces diffuse (left) and with a reflective box (right).

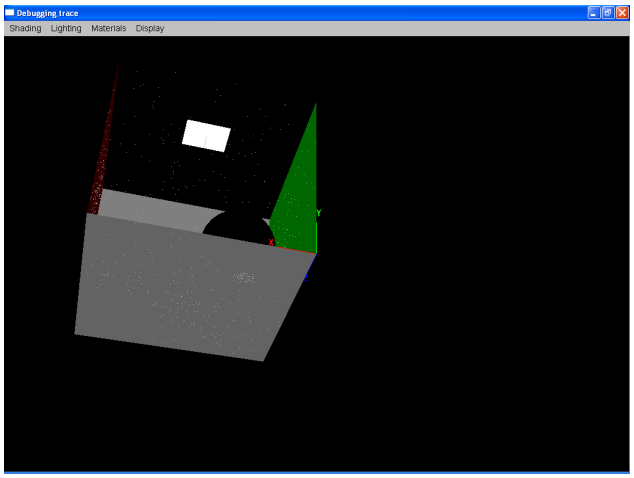

Figure 19: Debugging views of the caustic photon map with a transmissive sphere. Notice the caustic developing beneath the sphere.

3.5.3 Visualizing the global and caustic photon maps

Using k-nearest-neighbor matching the photon map provides an estimate of the irradiance at a point. Therefore, using photon maps one can obtain basic global illumination results, since the photons transfer irradiance at the various intersection points along their paths. The following images use the photon map estimates to render the scene. The code is in `material::shade`, when the `usePhotonMapEstimate` flag is set to true. (Fig.20.)

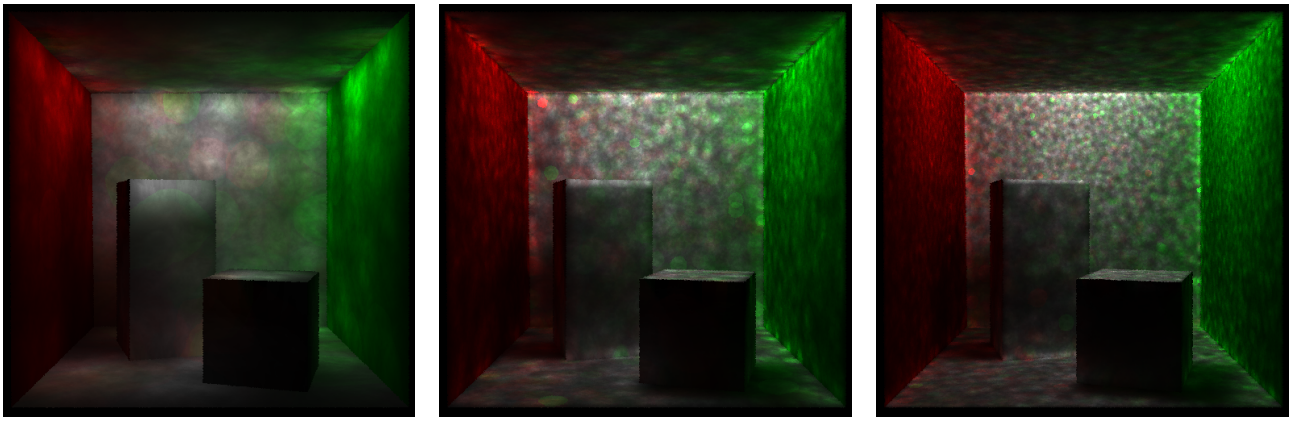

Figure 20: Rendering the cornell box using the global phton map with 10k, 50k, and 500k photons.

The patches of color are the effect of using the k-nearest-neighbor algorithm. As one increases the number of photons in the scene, for fixed number of photons used in the estimate, the patches become smaller, since smaller neighborhoods capture the same number of photons.

Similarly, one may use the caustic photon map to render the scene (fig.21).

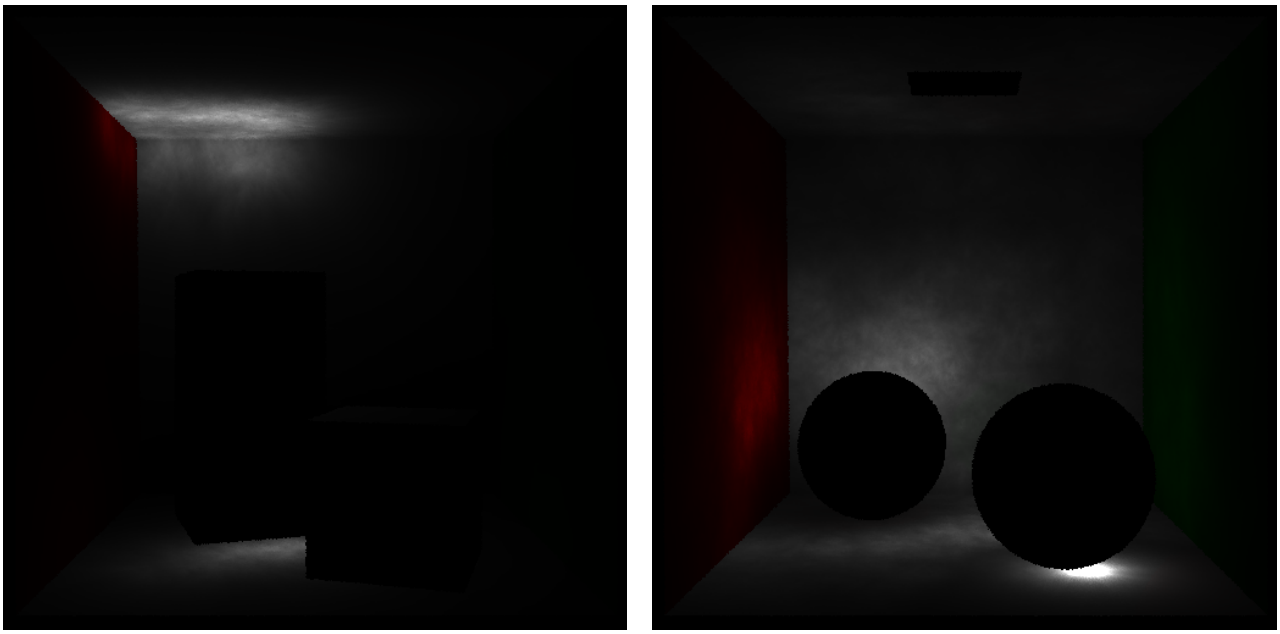

Figure 21: Rendering the scene using the caustic photon map. Cornell box with reflective boxes (left) and transmissive and reflective sphere (right).

Finally, one may of course use both the global and caustic photon map estimates to render a scene. The following series shows the use of the global photon map estimates, caustic photon map estimates, and both together. (Fig.22.)

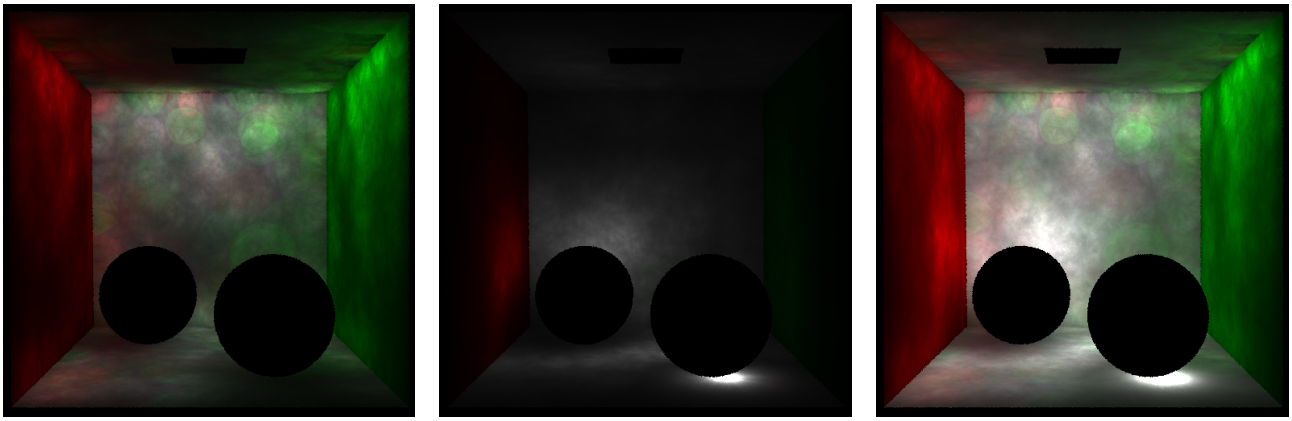

Figure 22: Rendering the scene using global, caustic, and combined photon map estimates.

3.5.4 Rendering the scene using the photon map estimates

The final phase of photon mapping is rendering making use of the global and caustic photon map estimates. Rays are sent and reflected/refracted as with regular path tracing, but now the irradiance at a point is obtained using the photon map estimate at that point (fig.23).

At this stage, the biasedness of photon mapping becomes an issue. The number of photons used in the global versus caustic maps, and the number of photons used in each estimate, have a strong effect not only on the intensity but the shapes of the lighting effects. Two scenes listed as extra credit on the CMU page are figs.24,25. The first involves reflection on cubes (rather than just spheres) and the second transmission/reflection using arbitrary surfaces (water). Both could benefit from being rendered for longer times to remove noise; these renderings took about 2-3 hours each.

References

[1] James Arvo, Phil Dutre, et al.. *Monte Carlo Ray Tracing.* Siggraph 2003, Course 44, July 2003.

[2] Henrik Wann Jensen. *Realistic image synthesis using photon mapping.* A.K. Peters 2001. (Beware lots of typos in critical formulas.)

[3] Matt Pharr, Greg Humphreys. *Physically based rendering.* Elsevier 2004.

[4] Peter Shirley. *Realistic Ray Tracing.* A.K. Peters 2000.

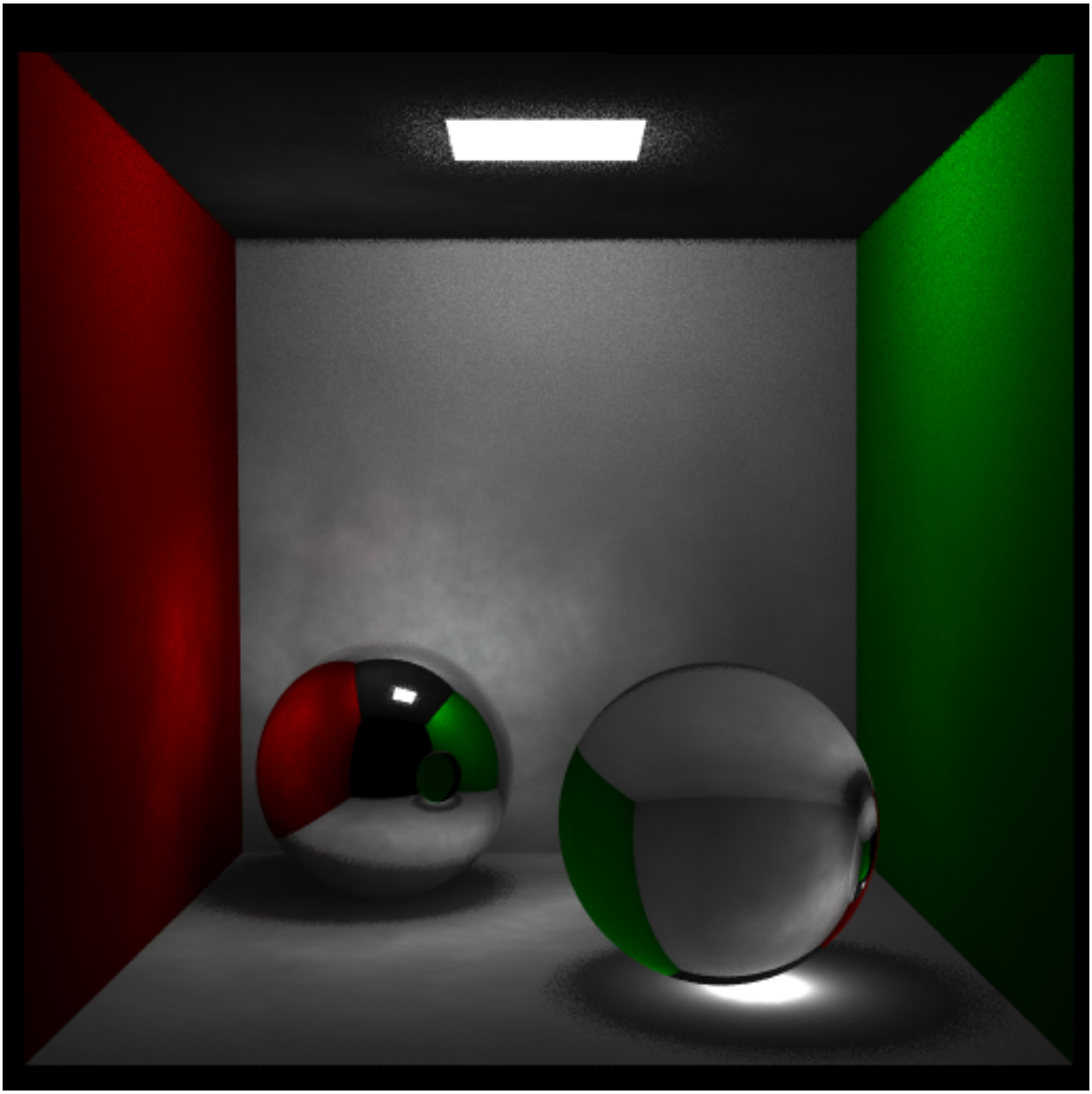

Figure 23: `cornellboxspheres.ray`. Path depth of 4, 4 samples/pixel edge, 44k photons in global map, 500k photons in caustic map, 88 photons/estimate.

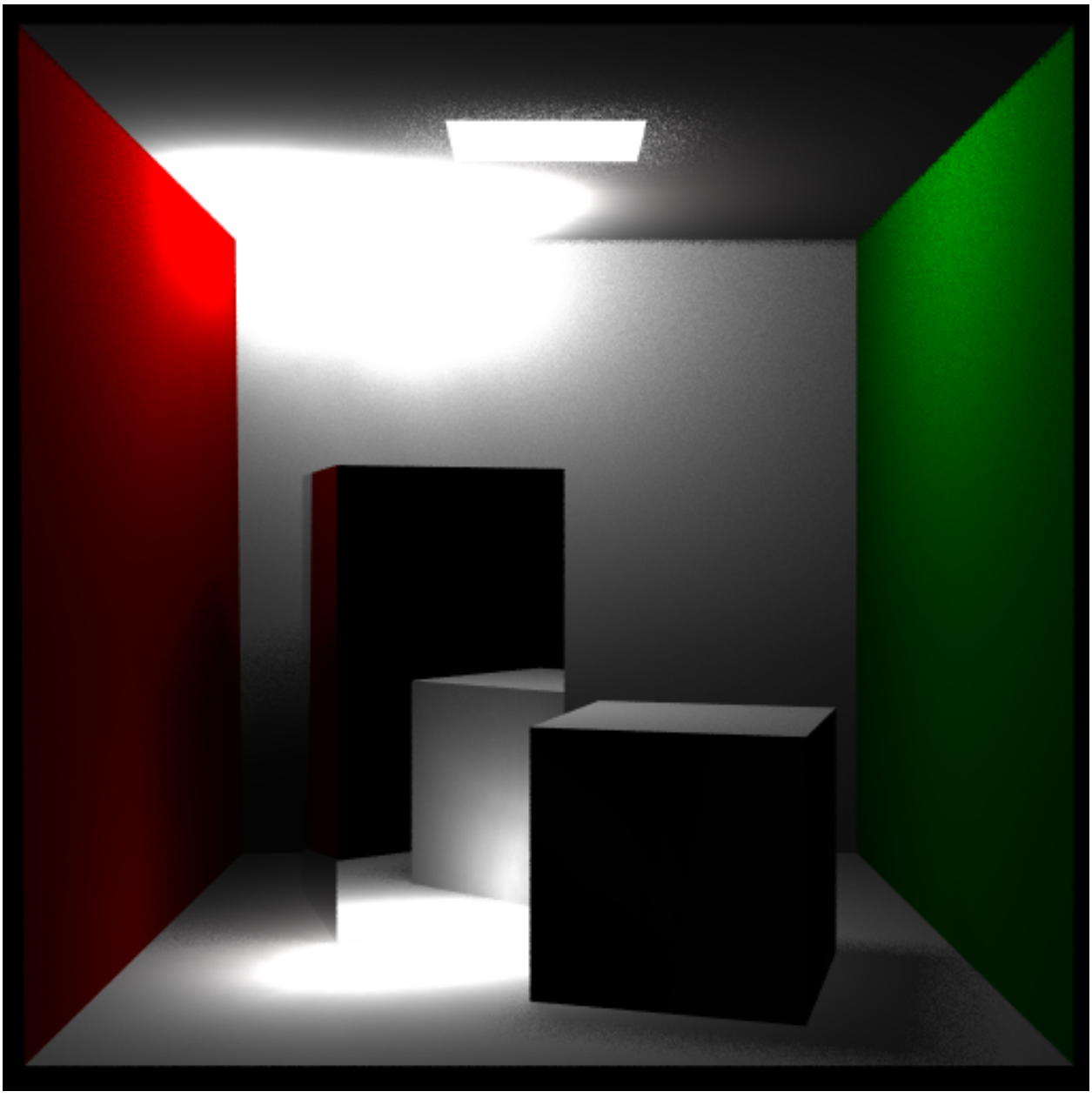

Figure 24: Photon mapping with reflective boxes. Probably too many photons in the caustic map—the bias of photon mapping is evident. Path depth of 4, 4 samples/pixel edge,604k photons in global map, 220k photons in caustic map, 445 photons/estimate.

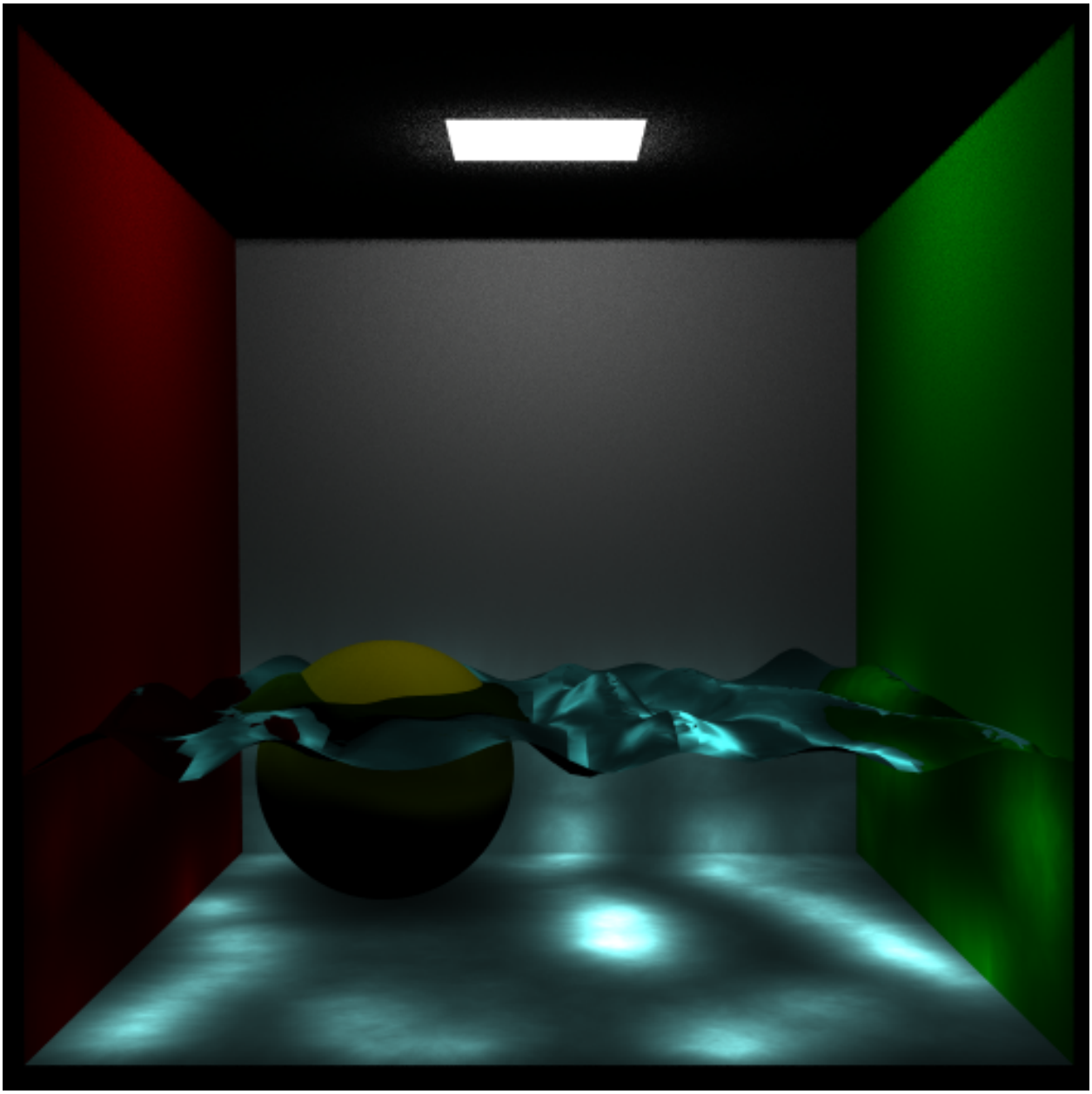

Figure 25: Photon mapping scene with water. Path depth of 4, 6 samples/pixel edge,440k photons in global map, 524k photons in caustic map, 120 photons/estimate.