Vaibhav Malpani vom2102
Guillaume Le Chenadec gpl2118
Rahul Tewari rt2520

PHOTON MAPPING

For this project, we chose to implement Photon Mapping. Photon Mapping is a rendering algorithm that, like Path Tracing, approximates Global Illumination. The method is different from Path Tracing in that instead of tracing the rays from the eye to the lights, we throw photons from the light and trace these photons into the scene to then use the intersections of the photons with the scene as the source of illumination. Moreover, it is computationally easier to render effects such as caustics, that we also implemented.

Photon Mapping was developed by Henrik Wann Jensen, and all the equations in our code are derived from his Siggraph Course [1]. We built our code on top of the code from PA4.

PHOTON MAPPING

Sampling the lights

Our code handles both point light sources and area lights (represented by arbitrary meshes). It also handles multiple lights.

We will throw many photons from the lights into the scene, in all possible directions and record the intersections of photons with the scene.

In the case of point light sources, the origin of the ray that traces the photon is located at the position of the light, and the direction is generated randomly from a uniform distribution over the hemisphere.

Area lights can either be spheres or triangle meshes.

We select a position on the surface from a uniform distribution. In the case of a triangle-based mesh, we first select a triangle with uniform probability based on their area. That is, the bigger the triangle, the more likely it is to be picked. And then we uniformly sample a point on that triangle. Then, we sample a normal

based on a cosine-weighted distribution. That is, the direction of the ray is more likely to be close to the normal, and very unlikely to be parallel to the surface.

Then we can throw the photon in the computed direction.

Photon Storing

We have a known number of photons (a variable set by the user) that we want to throw into the scene. Based on the different lights intensities, each light will throw a fraction of this number. For instance, if both lights have the same intensity, they will throw the same number of photons.

When a photon hits a surface, some of its energy is reflected and some is absorbed, depending on the material's properties. Some of the energy will be reflected specularly, some diffusely. To simulate that, instead of "splitting" the photon, we just reflect (diffusely or specularly) or absorb it based on Russian Roulette's probabilities computed from the material's properties. Since we throw many photons, this leads to the same result, and is more efficient. One slight disadvantage of this technique is that it has more variance than Monte Carlo techniques but its advantage is that it saves us the time we would have wasted following low power photons.

Our materials thus have two BRDF components, a diffuse one and a specular one. The sum of the diffuse and specular reflectance must be less that one (to conserve energy).

When a photon hits a diffuse surface, we store it in the photon map, with its energy, position of intersection and incident direction. Note that along its path, a photon can be stored several times. We set a variable that controls the maximum number of reflections that we allow.

Once we have recorded all the intersections, we build a KD-Tree based on the positions of the photons.

Caustic Photon Map

The photon map described above is the global photon map. We have also implemented the caustic map. The caustic map is used to better render the caustics.

We direct the rays from a light source towards the objects that have some specular component. We didn't implement a projection map, instead what we do is sample a direction from the light as before, and then check if this direction intersects a bounding box of one of the specular objects. If it does, then we throw a photon, otherwise, we sample another direction and iterate.

Besides scaling the photon power by a light source's relative intensity, we also need to scale it by the ratio of the number of rays that hit a specular object over the total number of rays generated.

At most one photon can be recorded on a path in a caustic map. A photon intersection in the caustic map is recorded if the first object intersected is specular. And we store the photon on the first diffuse object it hits after that.

We implemented refractions for our caustic map. Our dielectrics are perfect dielectrics, and we represent them by their index of refractivity. Thus the diffuse and specular components are null.

Because in theory a ray will be partly reflected and partly refracted, we again use the Russian Roulette to decide if a ray is reflected or refracted. We use Schlick's approximation to compute the probabilities (based on the reflectivity and transmittance of the dielectric). Since we compute the refracted direction according to Snell's (Descartes') law, this implementation also handles 'total internal reflections'. So we set a variable to limit the number of internal reflections.

Like for the global map, we build a KD-Tree for this map

Estimating the radiance

To find the incident radiance at a point, we find the K-nearest (this is a parameter of our algorithm) photons of this point in the photon map, within a sphere of specific radius (that is another parameter). Then we use these photons to approximate the incoming radiance at this position.

We use our balanced KD-Tree to make this fast. So we wrote a function that finds the K-nearest neighbors of a position using our KD-Tree structure. Then we use these photons to approximate the incoming radiance at this position, and

combine this incident radiance with the BRDF of our material to find the incident radiance.

Rendering the final image

We can now use this incoming radiance to compute our final image. Like for Ray Tracing, we throw our rays through the pixels, and find their intersection with the scene.

We handle refractions and reflections here as well. Basically, if a ray hits a mirror, it is reflected, and we find the radiance at the intersection of the reflected ray with the scene. When a ray hits a transparent material, if there is no refracted ray, it acts as a mirror. If there is both a refracted ray and reflected ray, we recursively find the radiance for these two rays and blend these two radiances using Schilck's approximation for reflectivity. We handle multiple internal reflections/refractions, and set a limit on the number of internal reflections.

When a ray hits a diffuse or specular material, we use its BRDF and the estimated incoming radiance to compute the radiance of this ray.

The results of photon mapping are a bit noisy, so we also implemented a modification of the previous algorithm where the direct lighting is computed via the classic ray tracing algorithm. We computed the direct lighting using LuminaireIlluminator, so as to have soft shadows in our direct lighting. Then we incorporate the indirect illumination from photon maps (global and caustic) to this direct lighting. Thus for this technique we don't take into account the photons that represent the direct lighting in the global photon maps.

MATERIALS

We have also added some new materials: "DiffuseAndSpecular"(Glossy), "Mirror"(Ideal reflective surfaces) and "Glass" (anything with refractive index greater than 1 is considered glass).

CONCLUSION

For this project, we focused on implementing a photon mapping algorithm that correctly implements all the physics and mathematics equations involved in photon mapping and ray tracing, so as to have a “physically” correct rendering, where we don’t need to use some “hacks” or pre-specified knowledge about the scene, by making our algorithms as general as possible.

RESULTS

All our images were computed in less than 15 minutes, even when throwing 10 millions photons for the global map, a few millions for the caustic map and using a jittered sampler of size 15x15. But we can have as good results with much less photons. In general, 2 or 3 millions photons give very good results and run in a few minutes.

The following parameters were adjusted in the different images:

- Number of global photons emitted
- Number of caustic photons emitted
- Maximum number of bounces of light
- K, where K is the number of neighbouring photons at a point used to estimate the irradiance at a point.
- Radius of the above mentioned neighbourhood

There are also different strategies to approximate the incoming radiance. When we search for the nearest photons and compute the incidant radiance based on their energy, in order to transform a flux to an incident radiance, we have to divide by the surface element on which we are looking for the photons. We can either approximate this area by Pi*r*r where r is the biggest distance to the K

nearest photons. Otherwise, we can divide each photon intensity by the square distance from the photon to the point.

The latter option is used here :

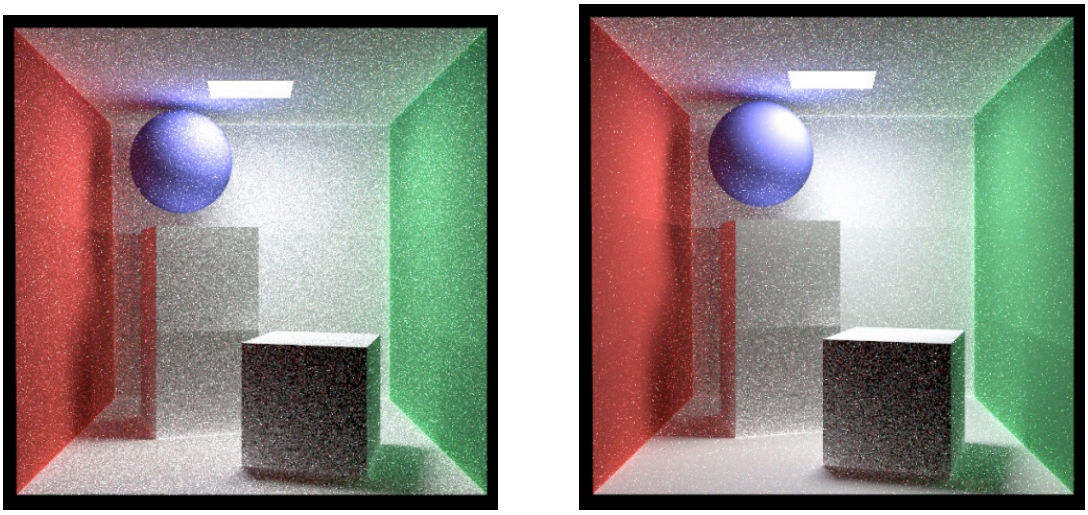


Fig1. A Cornell box rendered using Photon mapping (left) and Direct Lighting plus photon mapping (right)

In figure 1 we throw around 10000000 photons for the global map, and around 500000 for the caustic (the sphere has some specularity, using the caustic map increases the blue caustic). The indirect illumination is clearly visible in these images. We also sample each pixel by a 16 by 16 grid. These parameters are very big, but even with these parameters the running time was around 1000 seconds. It is possible to have as or better looking images with smaller parameters.

Here in fig 2 we have some images rendered with the second method, they look smoother, but there is some kind of pattern.

An important approximation used in the photon mapping algorithm is that the surfaces are locally flat, which sometimes leads to some errors (on the edges) as can be seen in the next images.

These two images were obtained in less than 12 minutes, throwing 5 millions photons into the scene for the global map, and one million for the caustic map.

We changed the intensity of the second image from 60 to 110 in each channel, that's why it is brighter.

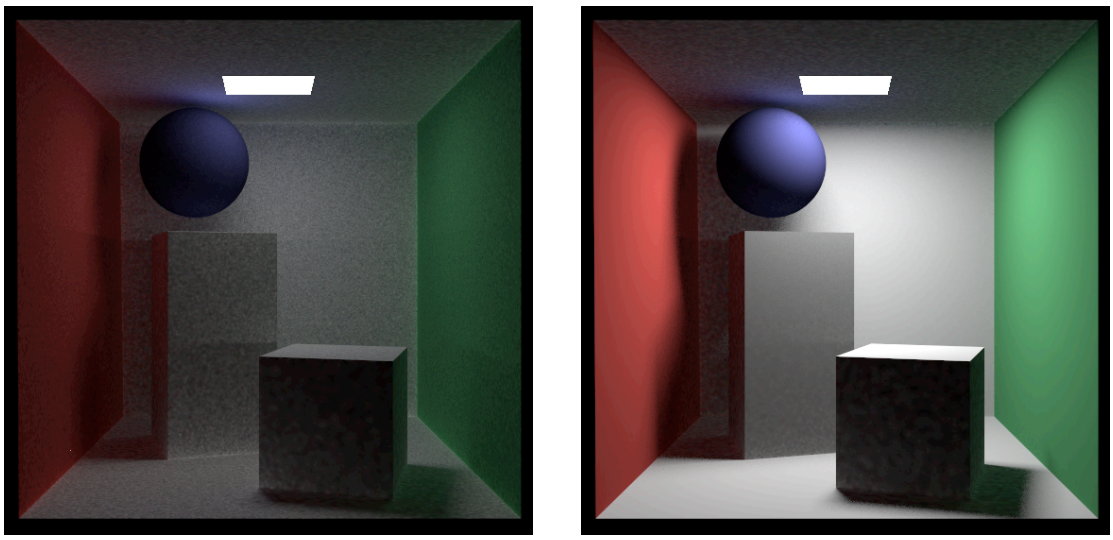
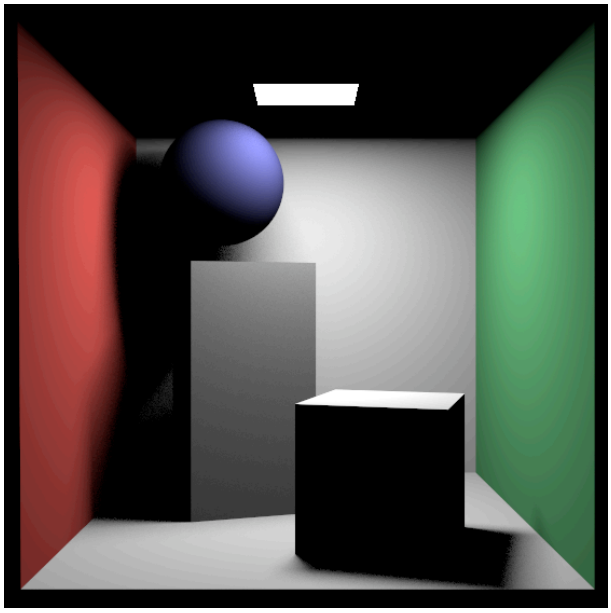


Figure 2: Still pure photon mapping on the left, and using direct lighting plus photon mapping on the right.

In comparison, the above scene looks like the following in the absence of global illumination:



Some examples of the caustic map:

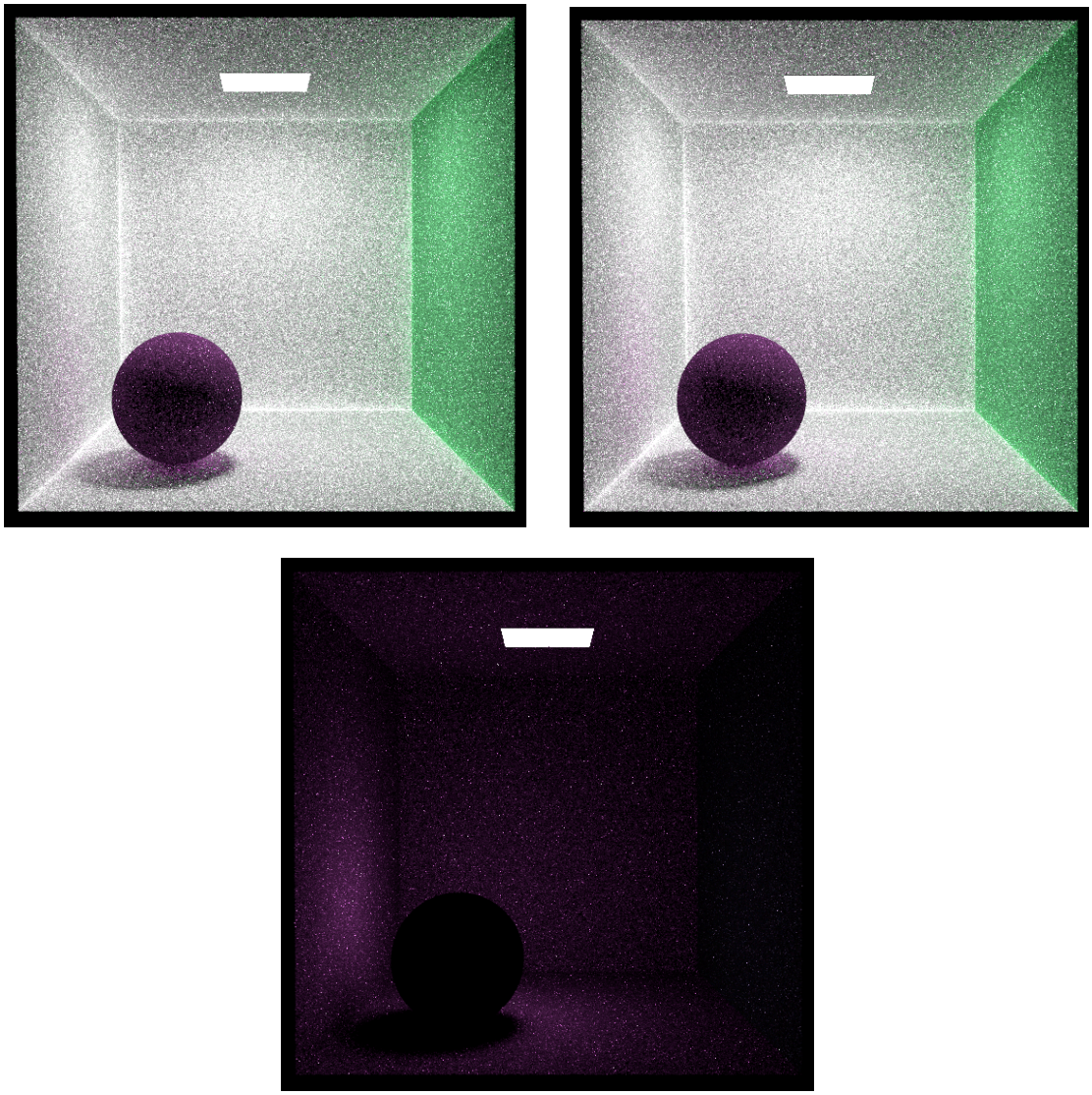


Fig 3: The Cornell box using just the global photon map(top left), using global and caustic photon map (top right)and showing just the caustic photon map (bottom)

The sphere has some specular and diffuse coefficients. On the top left we don't use the caustic map, but on the top right we do, so the caustic is slightly more visible. We show the colors obtained only from the caustic map on the last image. The specular color of the sphere is magenta.

Trying different parameters for the number of photons, K, the search radius etc may give better results.

Results with refraction and reflection:

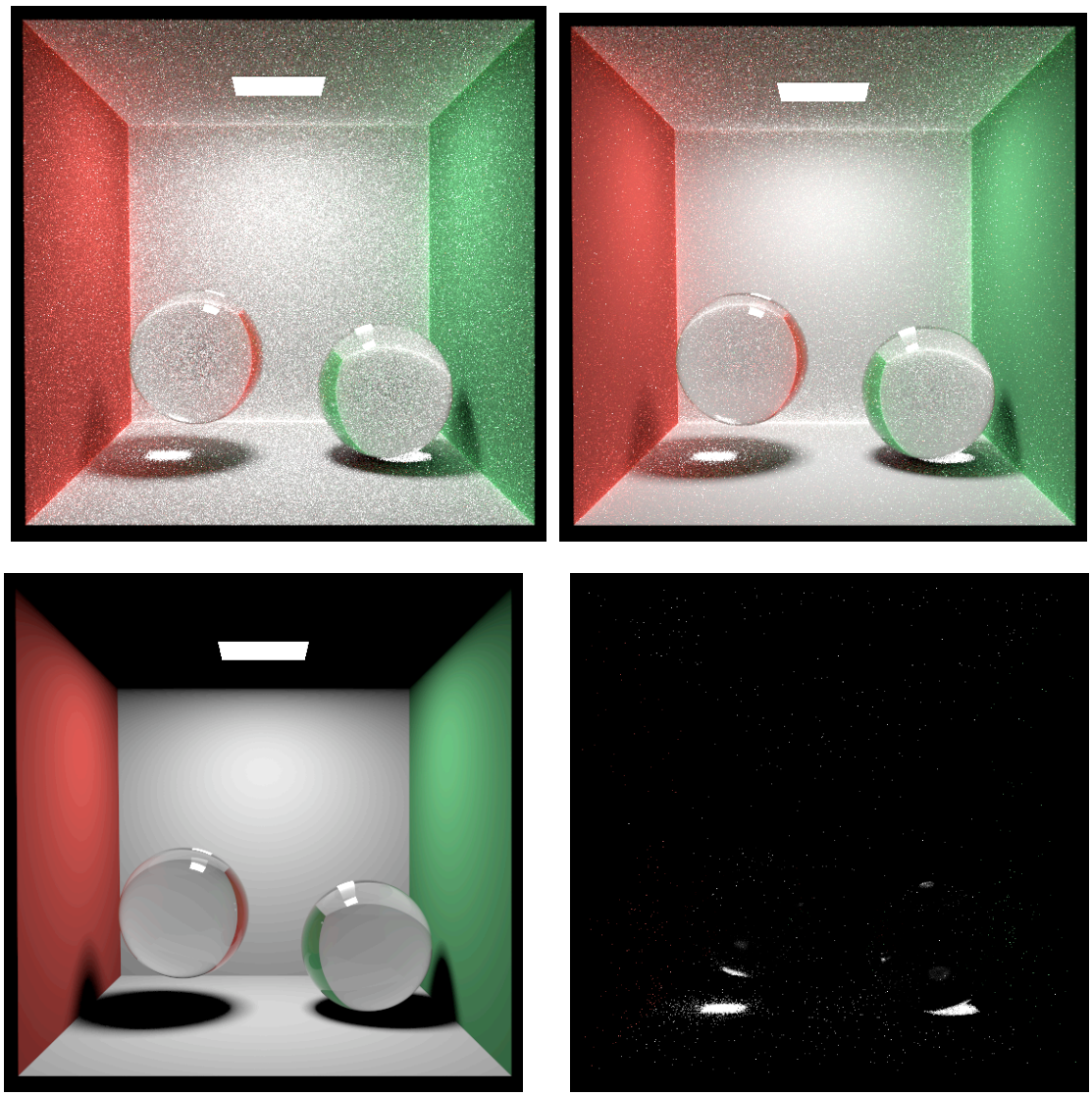


Fig 4. Using Photon mapping (top left) and Direct Lighting plus photon mapping (top right). Direct Lighting (bottom left) and caustic map (bottom right)

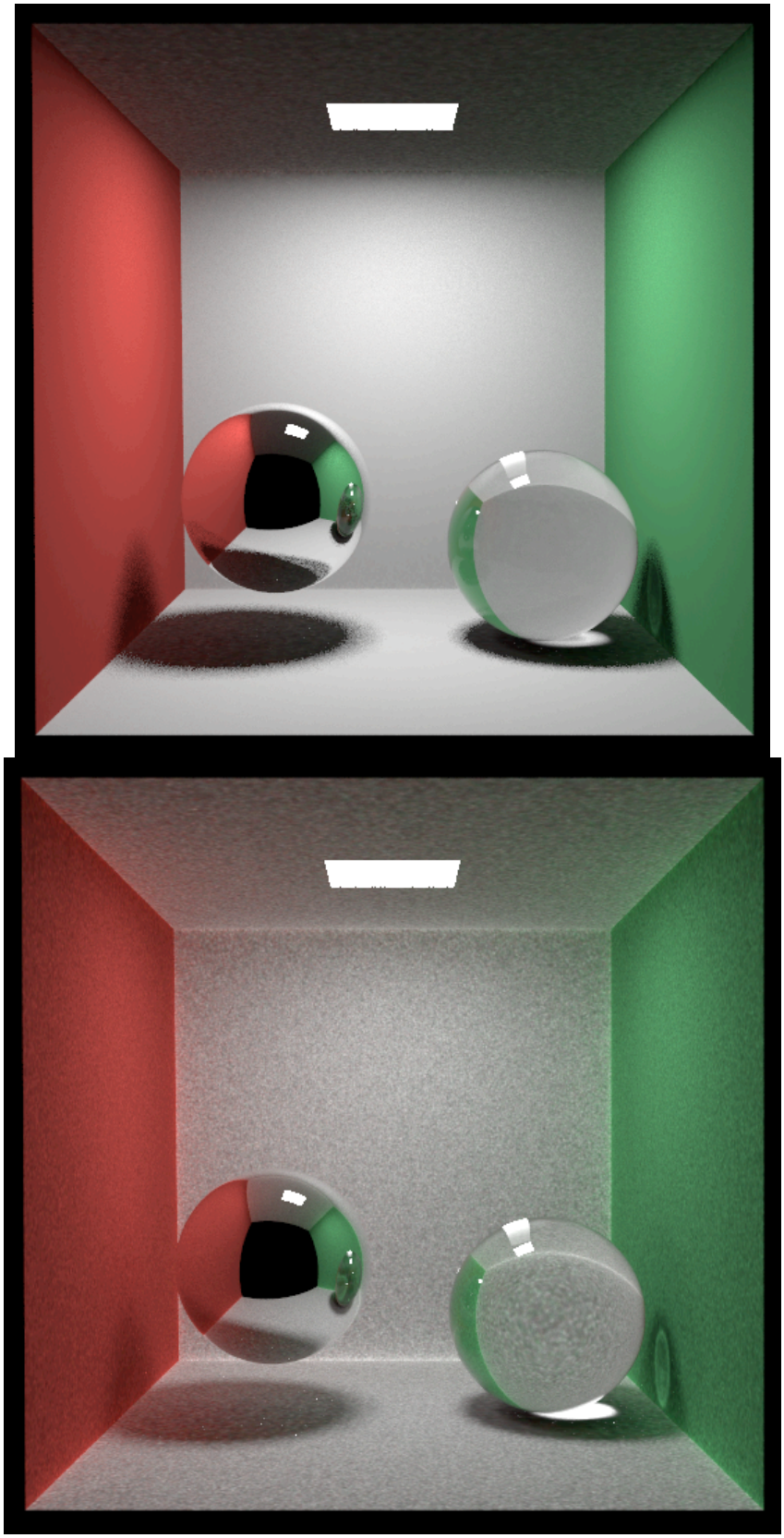


Fig 5 : Direct Lighting plus photon mapping (left) and photon mapping only (right) for a Cornell Box with a reflecting and a refractive surface.

Using several lights of different colors

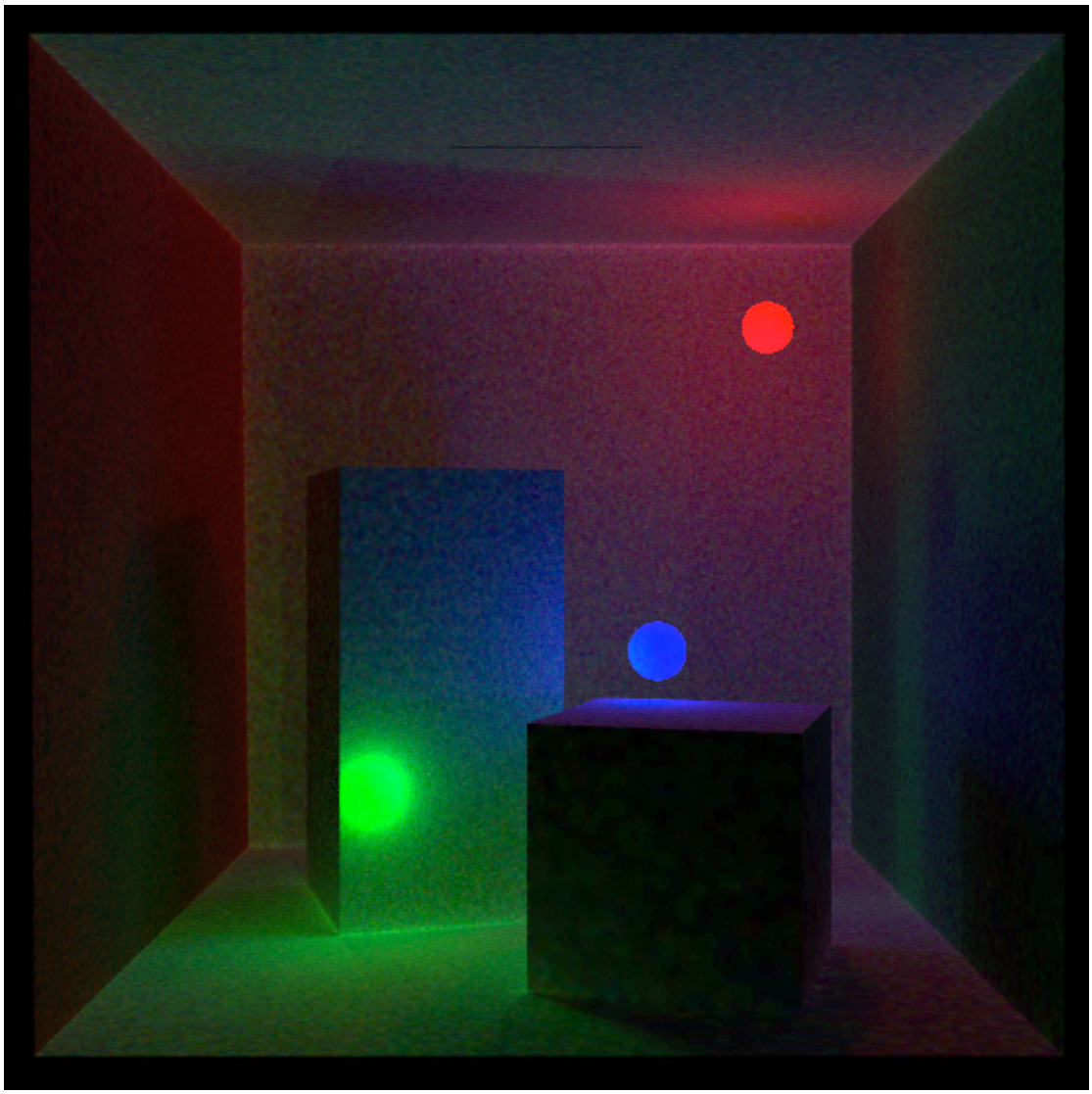


Fig 6: Using photon mapping with several lights

More refraction:

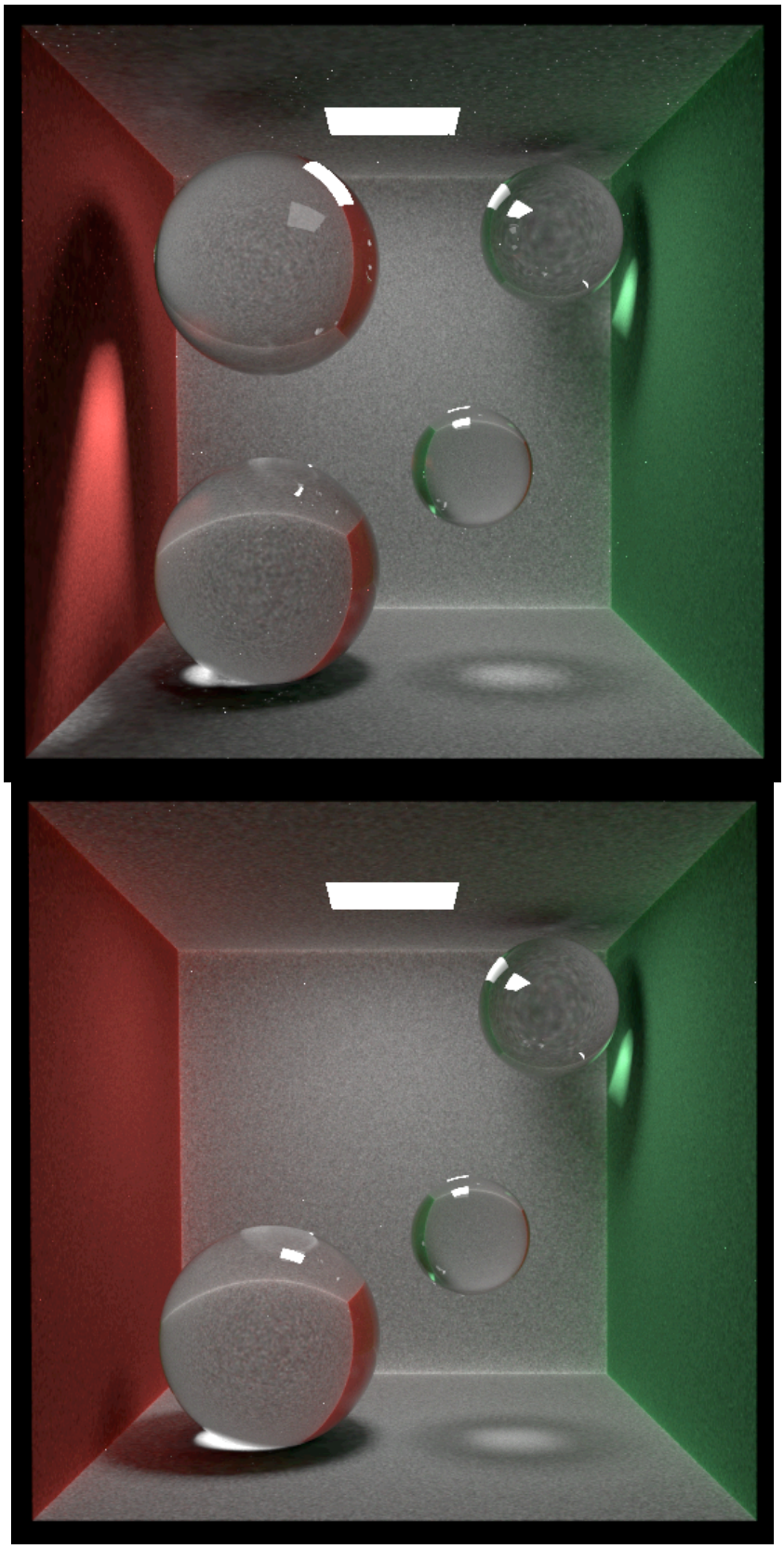


Fig 7: Using photon mapping only, 7 millions photons for the global map, 1 million for the caustic map. In 463 seconds.

A mistake that happened, we put the camera and the whole scene inside a transparent sphere by error (using photon mapping), this guy showed up and it looked cool so we included it:

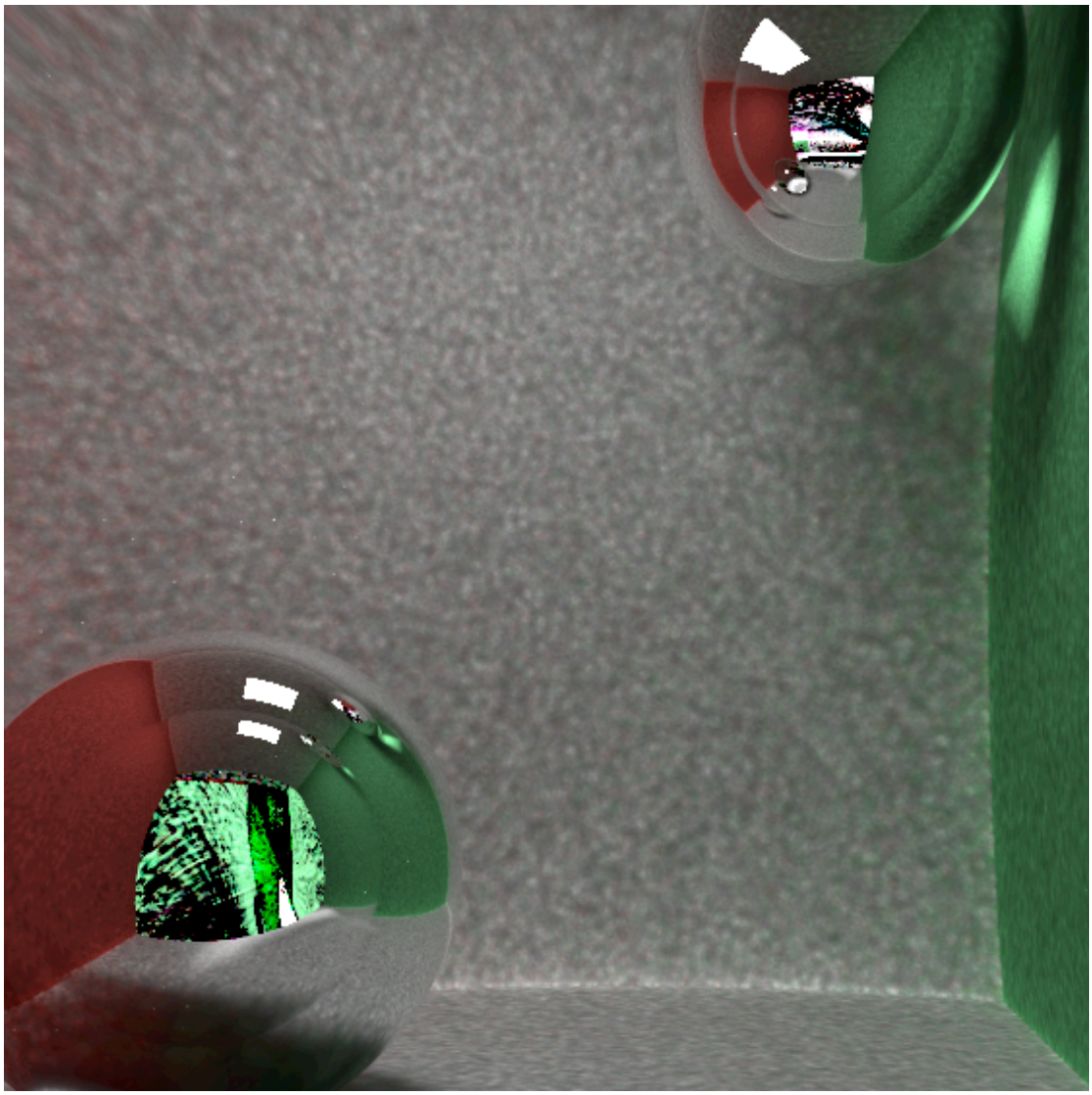


Fig 8: The scene and camera where included in a glass sphere by mistake

REFERENCES:

[1] H. W. Jensen and P. H. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps," in In Proceedings of SIGGRAPH'98, Orlando, 1998, pp. 311-320.

[2] Jensen, Henrik W., Realistic Image Synthesis Using Photon Mapping, A K Peters, Ltd., Massachusetts, 2001