CSE 306 Ray Tracer - Merlin Beaufils

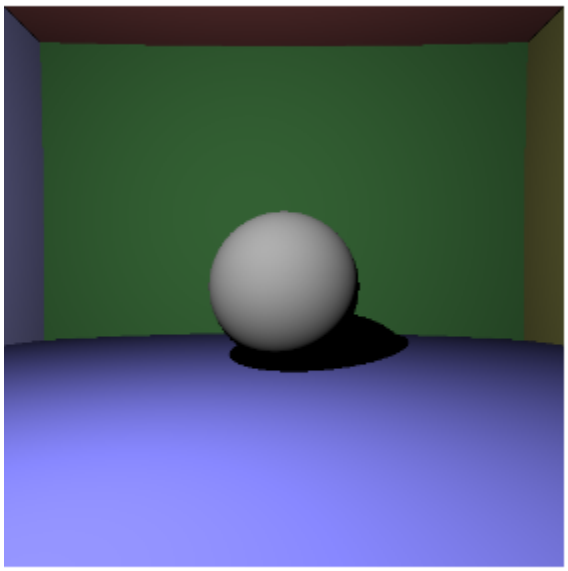
Introduction:

In this project we set out to build a ray tracer using relatively simple intersection algorithms and simulating non-deterministic algorithms by sending multiple rays per pixels. My code quickly becomes slow with more than 10 rays per pixel so we do not go much further than this.

We find it easier to handle spheres so we will mostly use those. We can simulate planes for the walls, floor and roof by using large enough spheres. We then try to improve our ray-tracer by using different lighting techniques and implementing mirrors, reflections and hollowness to our spheres.

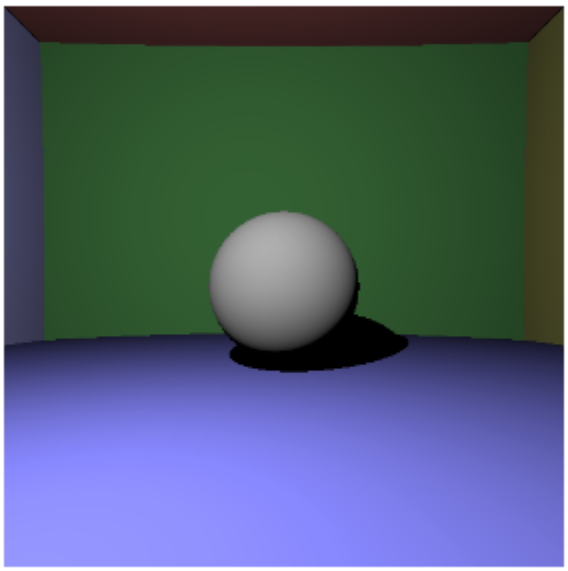
We start by building a simple scene:

1 Ray/Pixel



~6

10 rays/ pixel



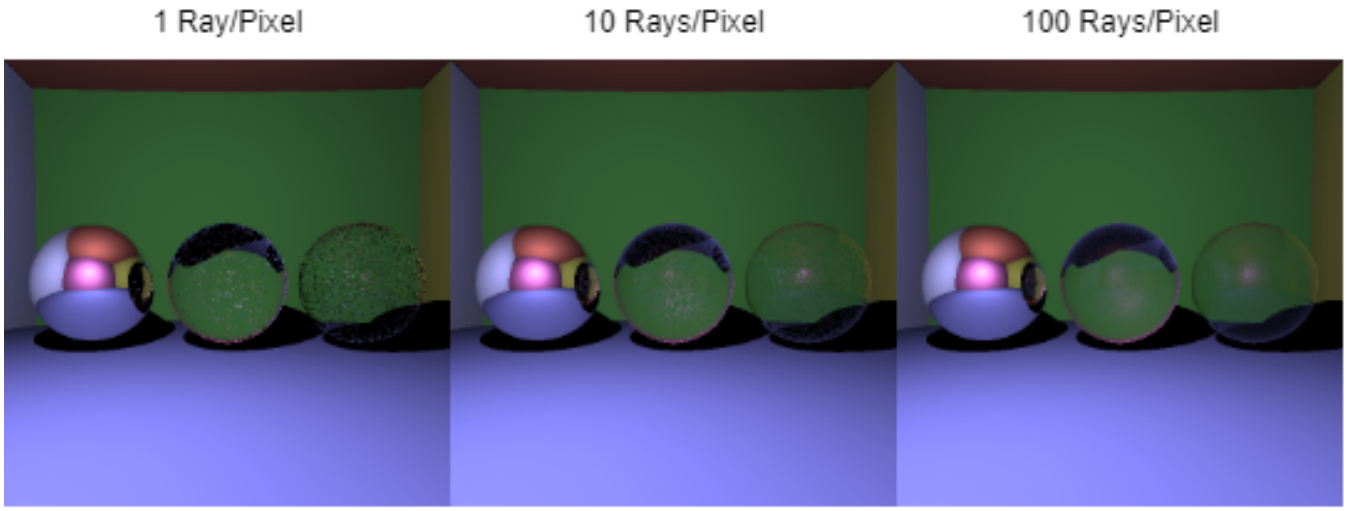
~30s

As expected there is no difference when we launch 1 ray per pixel or 10 rays per pixel. So far the algorithm used are deterministic. The time probably increased linearly given error for build time.

Mirrors, reflections and hollowness:

Using a Fresnel implementation, we implement our first case of randomness of the project for the sake of simplifying computation. When implementing a reflection, we cannot simply have

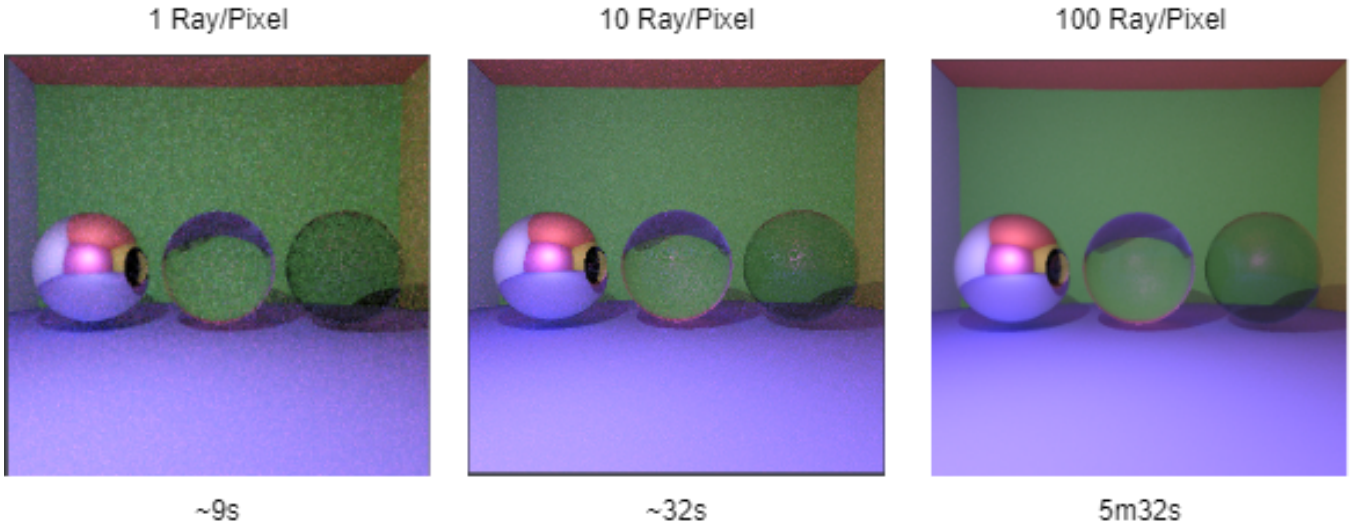
2^n rays in our simulation hence we stochastically keep one.



The diagram becomes a lot smoother with 100 rays per pixel but takes quite a bit more time. We see that when there is just one pixel, we get quite noisy results. Now let's deal with these shadows.

Indirect lighting:

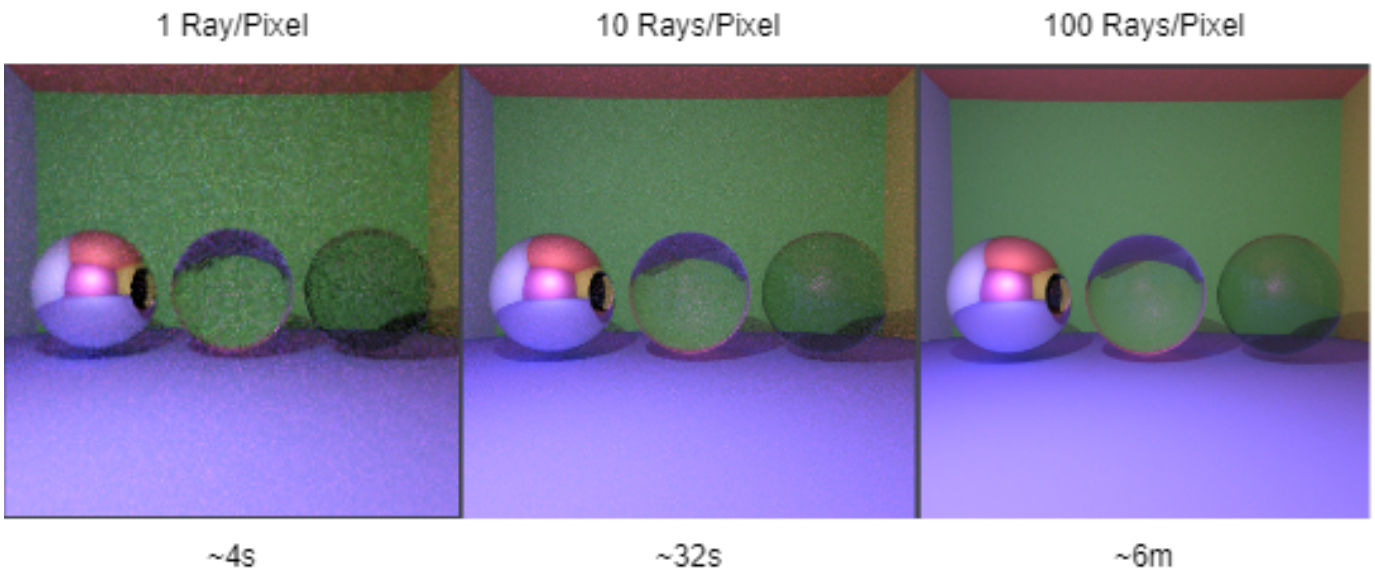
Now to deal with these unnatural contrasts in color, we will implement Indirect Lighting. I am directly using a spheric light source as I had some trouble with the code.



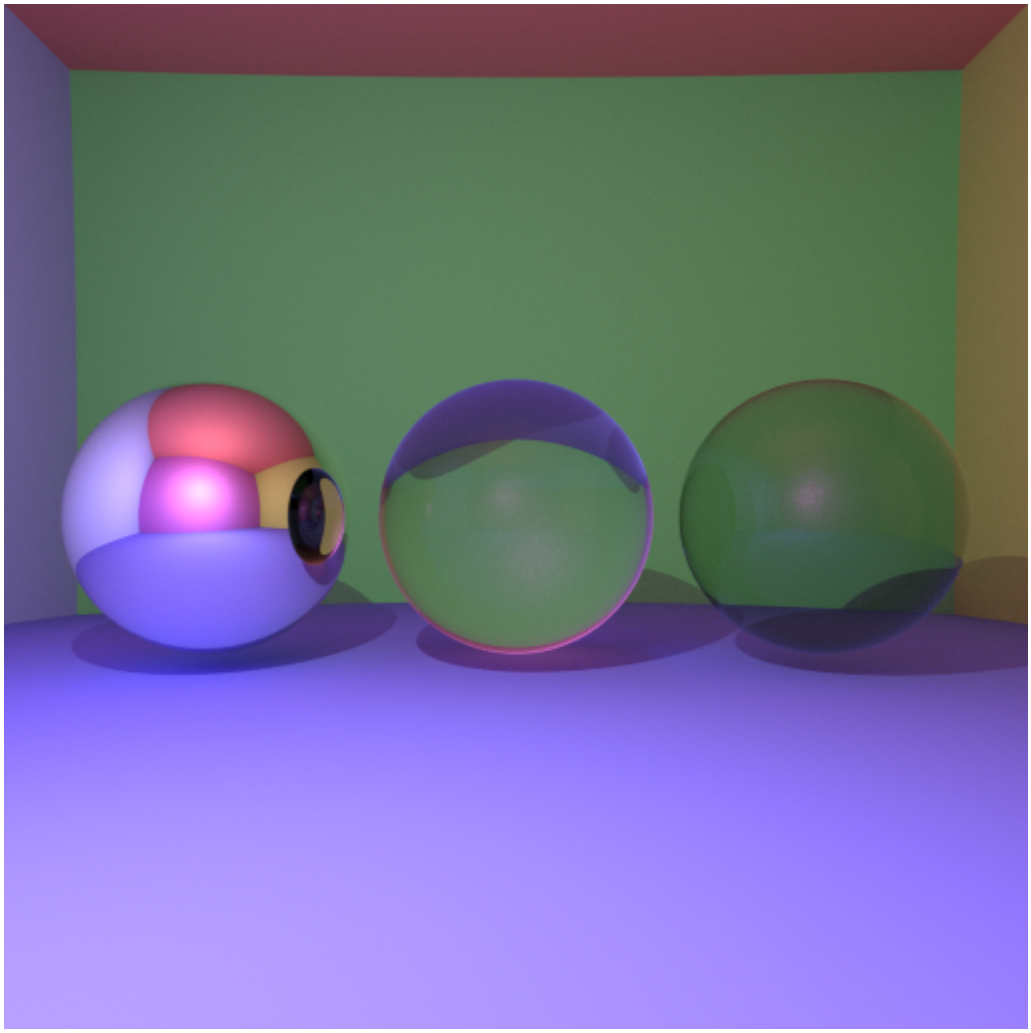
This code is starting to be very long. However it is worth it, as the shadows give a more natural look by stochastically blending the colors of the surroundings. It does also give a paler look to the photo that I am not such a fan of compared to the crisper direct lighting picture.

Anatialiasing:

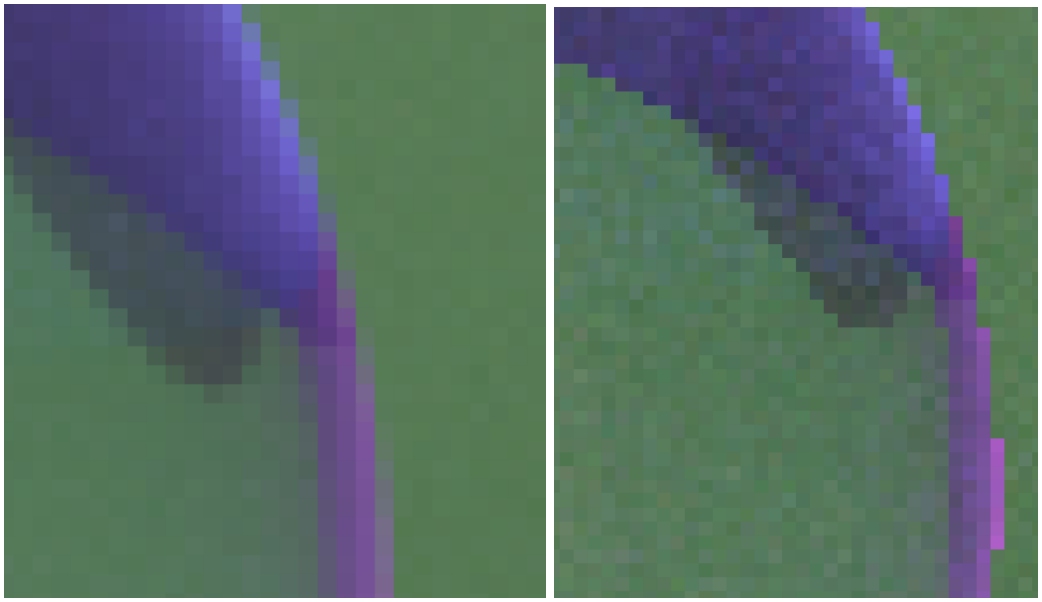
Let's move on to dealing with the contours of shapes. Because of our number of pixel restriction the previous images could look very staggered on the edges of circular shapes. With Anatialising we stochastically blend the pixel's area together to make a smoother image.



This is hard to see so I will show the resulting image for 1000 rays per pixel. This took 50 minutes to render. Very Smooth!



Let’s zoom in and compare this with the Indirect lighting render.



On the left, we have the Anatialiasing with a smooth contour and on the right, a pixelated edge.

Blurring:

We finally mess around with some blurring. Here I give the center sphere a horizontal motion of (1,0,0).