CSE306 - Computer Graphics

Raytracer

Tarcisio Soares Teixeira Neto

Computer Science
L'École Polytechnique
Palaiseau, France
May, 2021

1 Introduction

In this report I present the first assignment of the course CSE306 - Computer Graphics. This first assignment we studying ray tracing and how one can produce images. The main idea is to physically simulate how a camera would interact with an scene to take a photo. Throughout this project, we used several laws and formulas from physics, in order to be able to make a precise simulation.

In order to produce an image, we need to define first the scene that will be captured. For this project we consider the general set up as described in Fig 1. The scene will be bounded by four spheres, we assume the observer is at position $(0, 0, 55)$ looking towards direction $-\hat{z}$. Moreover, we consider a light centered at $(-10, 20, 40)$ (except Fig 7).

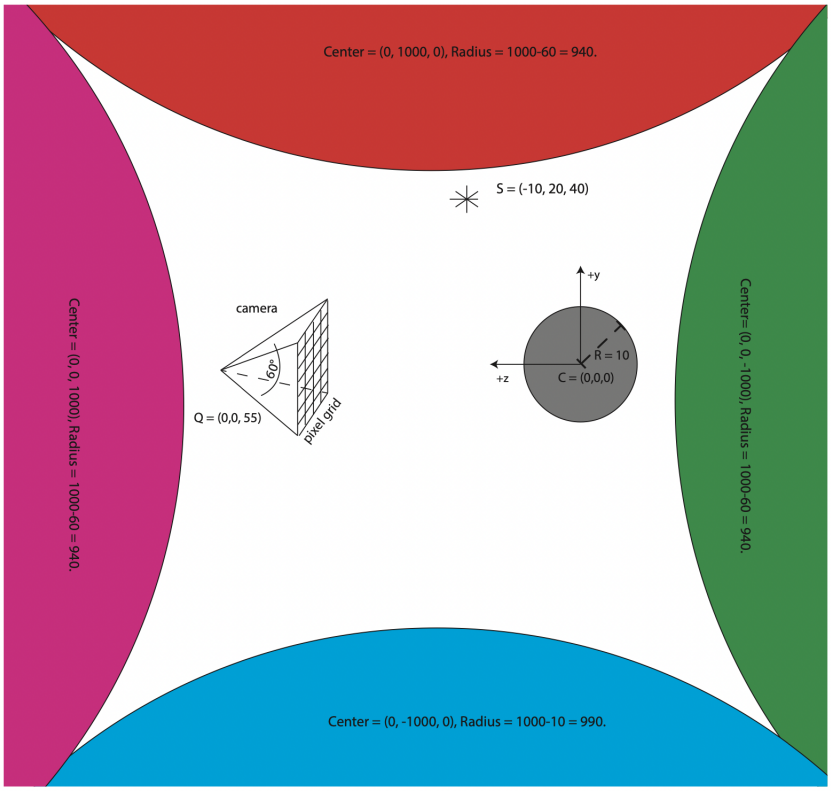

Figure 1: Figure taken from the lecture notes

2 Diffuse, reflective and refraction surfaces

In the first part of this project we study different types or surfaces and how the light and the camera rays interact with them. As mentioned in the lecture notes Computer screens do not react linearly with the pixel intensities they are fed with, thus we made use of an artifice *gamma correction* that consists in elevating the color retrieved to the power of $1/\gamma$, which in our case $\gamma = 1.22$. The result can be seen in Fig 2. This gamma correction will therefore be used throughout the entire project.

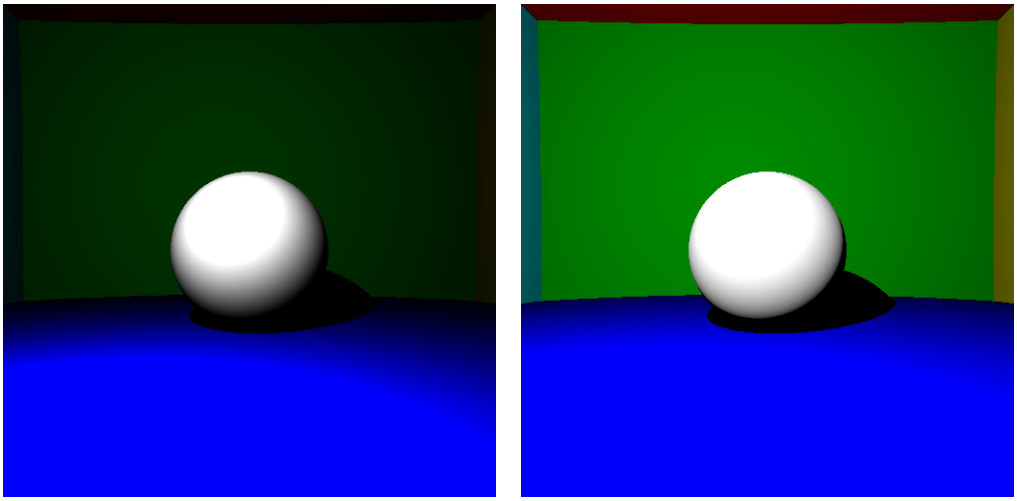

Figure 2: Image of white sphere with and without gamma correction. (left) Image produced without gamma correction and using point Light (rendering in 168ms without parallelization). (right) Image produced using gamma correction and point Light (rendering in 147ms without parallelization)

Besides diffuse surfaces we have seen the interaction of light and camera ray with refracted and reflective surfaces using laws from physics (e.g snell law). Furthermore, in order to produce accurate images we need to define a notion of ray depth, that is how many ray bounces we will allow. That notion clearly changes the picture. For example not allowing bounces the reflective surfaces would not reflect the light, similarly refractive surfaces needs multiple bounces to allow the light go through the surface. In Figure 3, we can see that in the left picture the reflection does not match the expected generating the black part (black is the default color) in the leftmost sphere in the LHS.

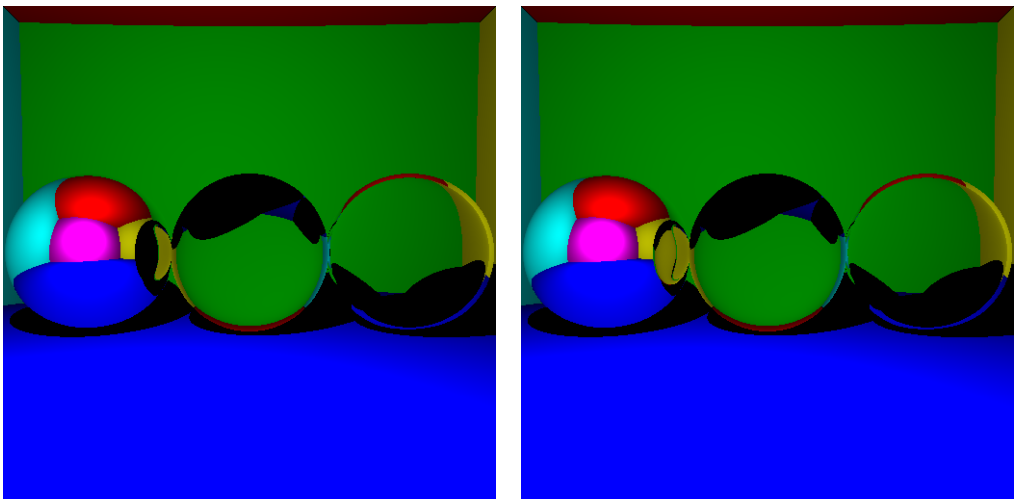

Figure 3: These are images of three spheres, one reflective, one refractive and another a hollow sphere. (left) using depth equals 6. (right) using ray depth equal 7. (each rendering takes around 265 ms without parallelization)

In addition, as refractive surfaces also reflect light using the Fresnel law we can obtain more accurate pictures as one can see in Figure 4.

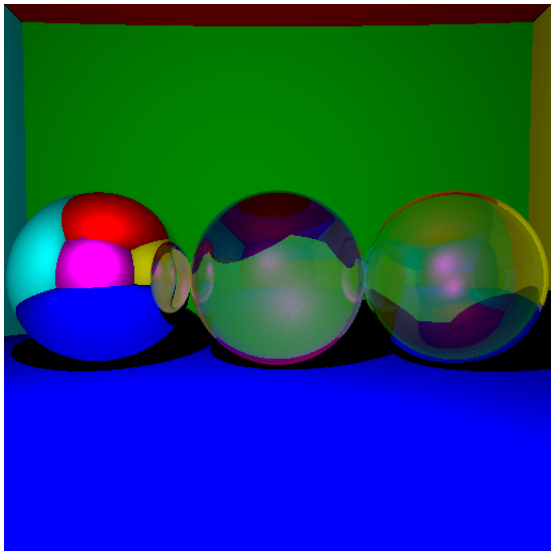

Figure 4: Images of three spheres, one reflective, one refractive and another a hollow sphere. Using Fresnel law for ray depth 7 and 1000 rays per pixel. (rendering takes around 3m 40s)

3 Indirect Lighting, antialiasing

As we know from physics even diffuse surfaces reflect some light. For example the sealing of a room is not dark even though it does not get direct light. In Figure 5 we have a comparison between a picture with and without indirect lighting. In the previous images we send rays in a constant direction for each pixel (targeting the middle of the pixel). Thus producing an image where the contour is not smooth. In order to solve this issue we can send several rays with slightly different directions per pixel. The result is portrayed in Figure 6.

4 Spherical light, depth field and motion blur

Previously we were considering point lights, however in reality that is not the case. In Figure 7 we have a clear representation of an spherical light and how it changes the picture. All the following pictures are constructed using spherical lights. In reality, objects in different distances might have different appearances depending on the focal distance of the lens used in the camera.

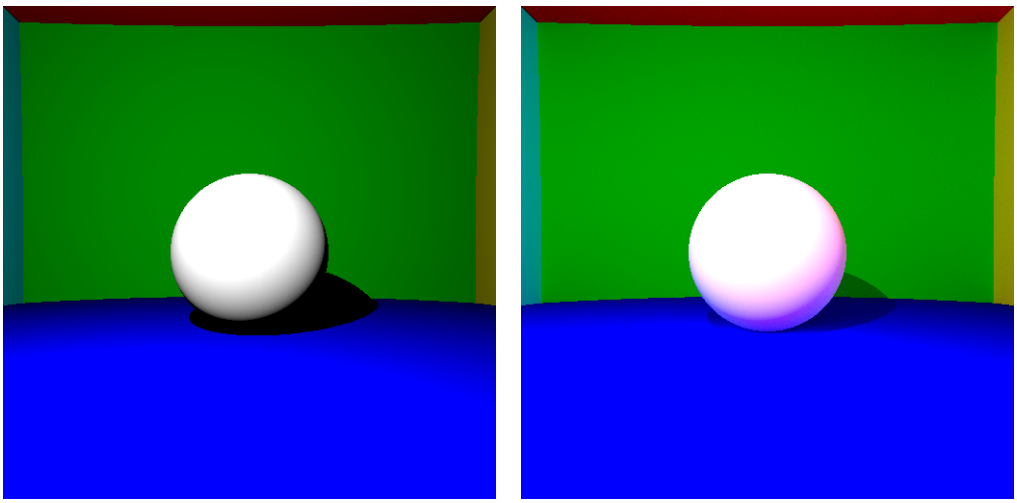

Figure 5: Comparison between an image with indirect lightening and without. (right) Picture implementing indirect lighting using 1000 rays per pixel rendering in 5m with parallelization.

Figure 8 portrays those differences.

Additionally, we can also simulate motion. Figure 9 portrays an object with some velocity.

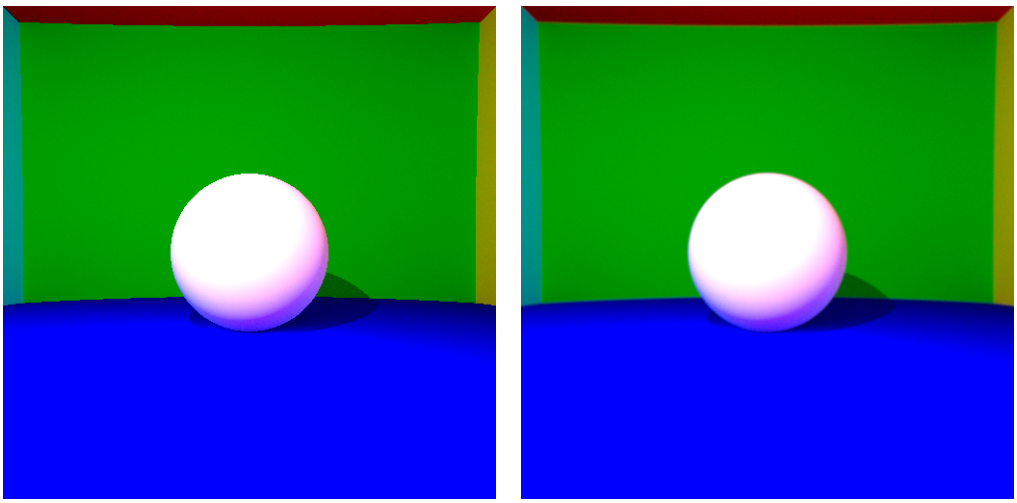

Figure 6: Comparison between an image with and without antialiasing mechanism. (right) Picture produced with antialiasing. (left) Picture produced without antialiasing. Both pictures with 1000 rays per pixel.

5 Meshes

In this section we see how the interaction occurs for meshes instead of only spheres. Meshes are briefly a set of triangles, therefore to get the intersection of a ray and a mesh it is necessary to find the exact triangle among all the triangles in the mesh that intersects the ray. When looking for the intersection of a ray and a scene containing a mesh we need to check if tha ray intersects with the mesh. This checking can be performed in several ways. First, one can check triangles one by one, which is very timing demanding (Figure 10 11 took around 20 minutes). One simple optimization is to add a box around the mesh and check if the ray goes through the box and we check the triangles only after. This simple idea reduces by a lot of the time (Figure 10 11 took around 3m 10s). Finally, one can consider multiple boxes instead and check the intersection of a ray with a mesh using a strategy similar to binary search in a tree of boxes. This approach reduces the time even more (Figure 10 11 took around 40s).

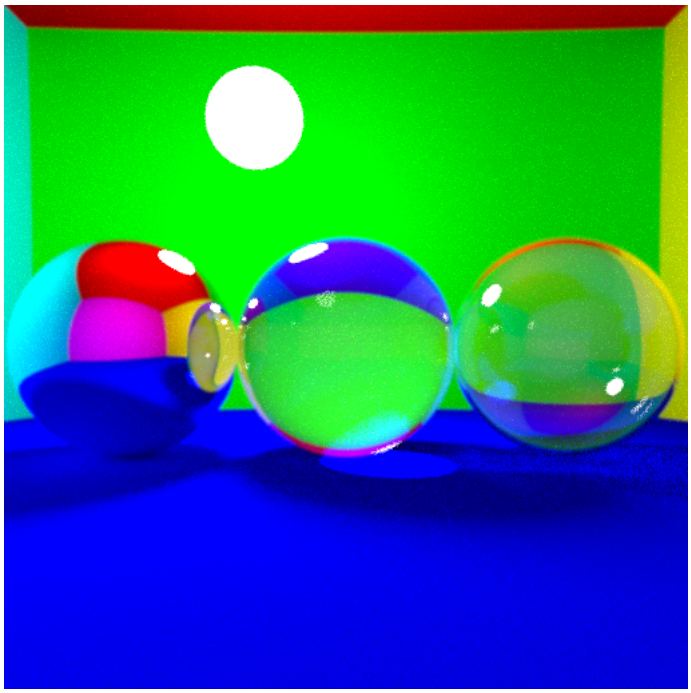

Figure 7: Images of three spheres, one reflective, one refractive and another a hollow sphere with the implementation of spherical light. This image was produced using 1000 rays per pixel and depth 7. (rendering in 10m with parallelization).

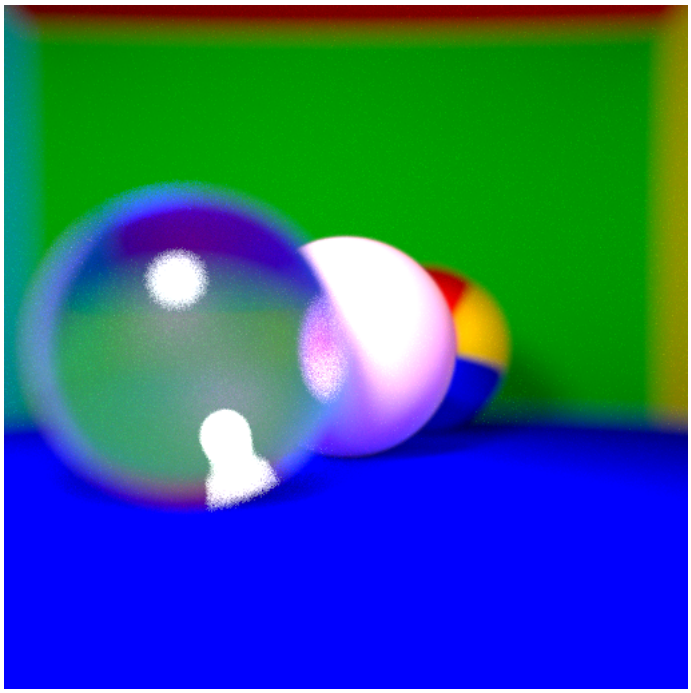

Figure 8: Images of three spheres, one reflective, one refractive and another white with the implementation of the notion of depth of field. This image was produced using 2000 rays per pixel and depth 7. (rendering in 15m with parallelization)

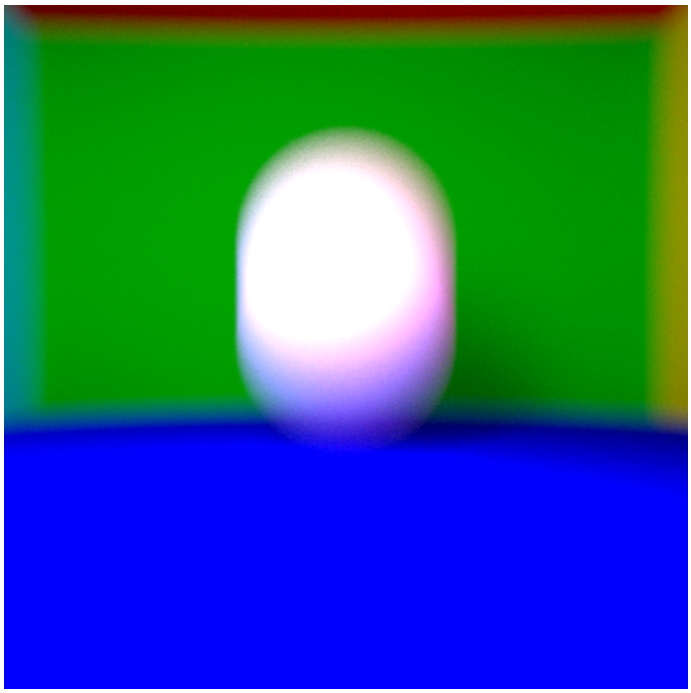

Figure 9: Image of a white sphere with the implementation of the notion of motion blur speed of 9.8. This image was produced using 1000 rays per pixel and depth 7. (rendering in 8m with parallelization)

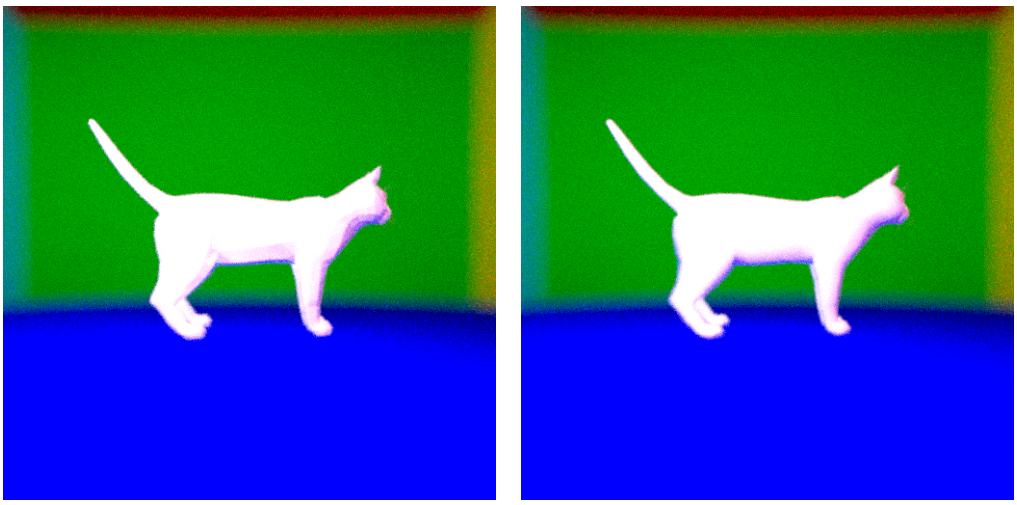

Figure 10: Image of a cat mesh. (left) Image with constant normals in each triangle of the mesh. (right) normal vector depending not only of the triangle but also the intersection point. Image produced with 32 rays per pixel

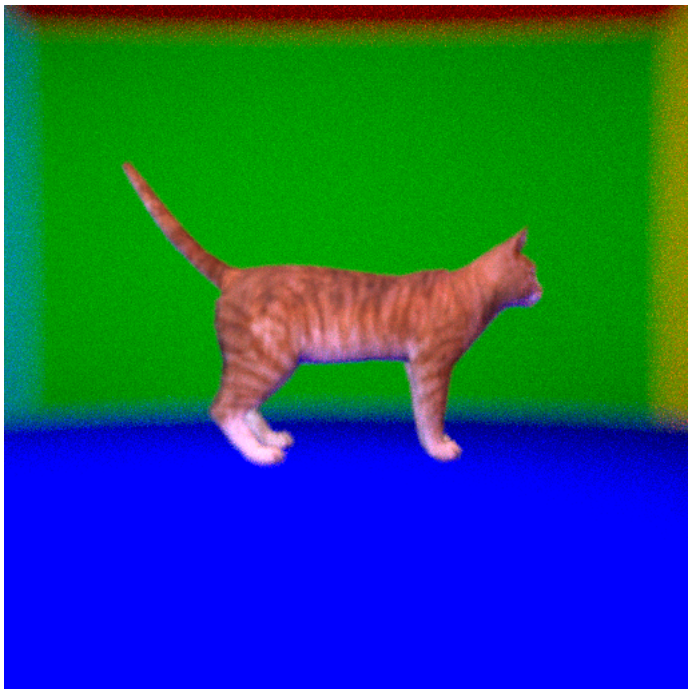

Figure 11: Image of a cat mesh with the proper texture. Image produced with 32 rays per pixel