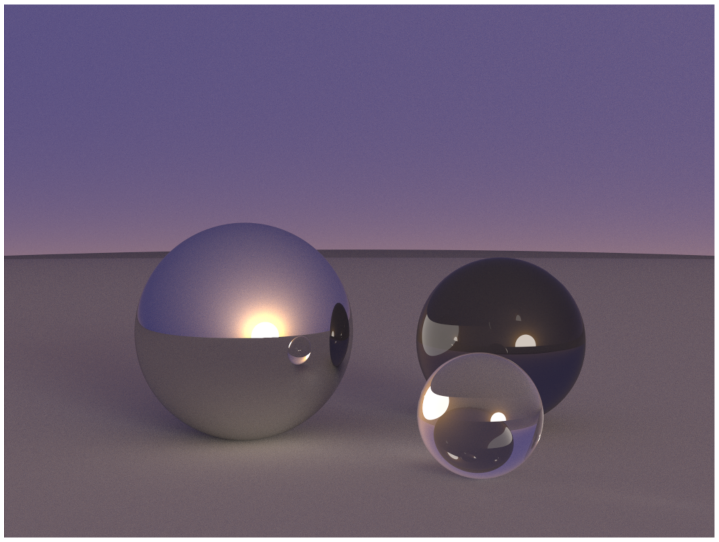
Master 1 IMAGINE

GASC Thibault

Ray Tracing Project : Step 1 and 2



University year 2022-2023

Contents

I Introduction

The ray tracing is used in 3D computer graphics. It is a technique for modeling light transport for use in a wide variety of rendering algorithms for generating images.

II STEP ONE :

1 Intersections with objects

A Ray/Sphere

To compute the intersection between the ray and spheres, I used the analytic solution. Let's consider the ray, which can be expressed using this function :

$$O + td \tag{1}$$

Where O the origin of the ray and D its direction, and the equation for a sphere which is :

$$x^2 + y^2 + z^2 = R^2 \tag{2}$$

Where x, y and z are the coordinates of a point and R the radius of the sphere. Let's consider that x, y and z are the coordinates of point P, we can write this following equation :

$$P^2 - R^2 = 0 \tag{3}$$

Now we can substitute equation 1 in equation 2 in order to replace P by the equation of the ray. So now, we have this equation :

$$(O + td)^2 - R^2 = 0 \tag{4}$$

By developing, we found a quadratic function with t as the unknown:

$$D^2t^2 + 2Odt + (O^2 - R^2) = 0 \tag{5}$$

Now that we have this equation, we can compute the solutions of this equation in order to find the intersection point.

To illustrate that, let me show you by using a table.

Δ	$= 0$	> 0	< 0
t	$= \frac{-b}{2a}$	$t_1 = \frac{-b-\sqrt{\Delta}}{2a}$ and $t_2 = \frac{-b+\sqrt{\Delta}}{2a}$	t is undefined
intersection	one intersection	two intersections $\rightarrow$ keep the min	no intersection

Table 1: Table representing the different solution

Then, to compute the normal of the sphere, we have to compute the intersection point (with the t that we have computed) and to normalize the vector formed by the intersection point and the center of the sphere.

Afterwards, in the function which is called *computeIntersection()*, we browse the sphere array in order to find the nearest intersection. When we have this intersection, we get the index of the object that we have intersected, we specify the type of intersect object, namely sphere or square. And then, in the function which is named *rayTraceRecursive()*, we have to change the color of the objects by using this variable of type of *Vec3* : objects[index]*.material.diffuse_material*.

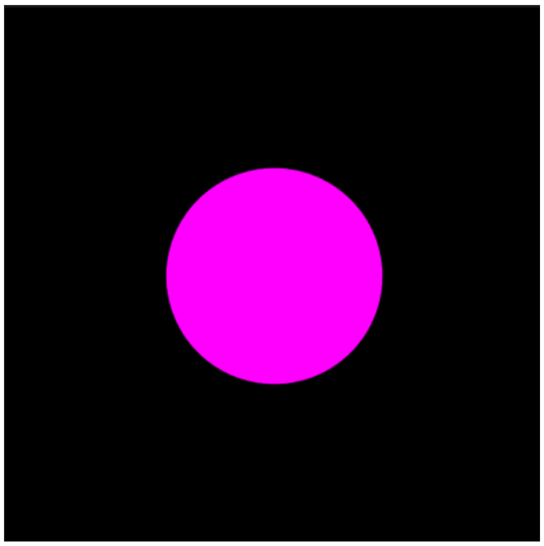


Figure 1: Result of ray tracing on sphere

B Ray/Square

For this intersection, we need the equation of a plane, namely :

$$x \cdot n - D = 0 \tag{6}$$

Where n the normal, and D the distance between the plane and the center (0,0,0), which is defined by $D = a \cdot n$. If we replace x by the equation of the ray (cf. equation 1), we get :

$$(O + td) \cdot n - D = 0 \tag{7}$$

And then, by rearranging the terms, we get :

$$t = \frac{D - O \cdot n}{d \cdot n} \tag{8}$$

To illustrate that, I will also use a table.

	intersection
$t > 0$	intersection in front of the camera
$t < 0$	intersection behind the camera
t undefined	ray confused with the plane
t infinity	ray is parallel and distinct from the plane

Table 2: Table representing the different t

And then proceed as for the spheres, i.e, we get the index of the object, the type of the object (sphere or square) and we change the color.

After all that, we can apply these two calculations on the Cornell's box.

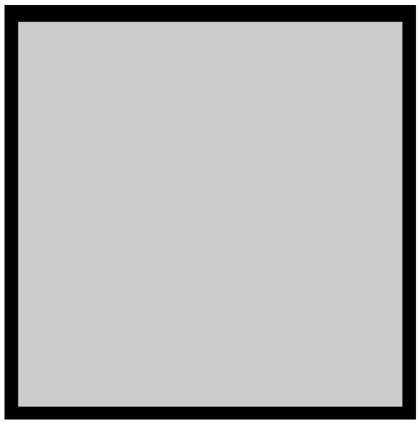


Figure 2: Result of ray tracing on square

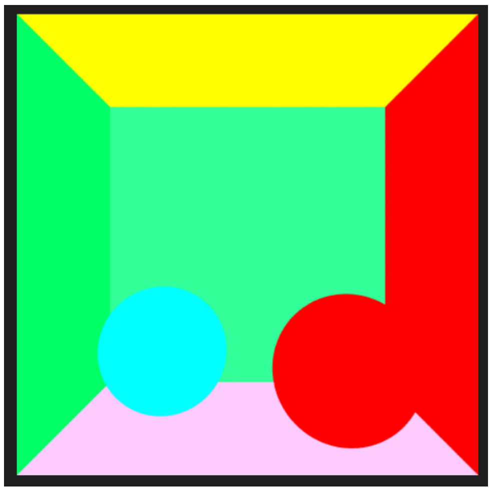


Figure 3: Result of ray tracing on Cornell's box

III STEP TWO :

1 Phong illumination

Now that we have display the Cornell box, we have to add Phong illumination. To do that, we have to use the following formula :

$$I = I_a K_a + \sum_{l}^{nb\_light} I_{ld} K_{ld} (L_l \cdot N) + I_{ls} K_{ls} (R_l \cdot V)^n \tag{9}$$

Where :

- $I_{a/d/s}$ is the intensity of the ambient/diffuse/specular light
- $K_{a/d/s}$ is the reflection coefficient of ambient/diffuse/specular light
- N is the normal
- L is the vector between the intersection point and the light position
- R is the direction of light reflection
- V is the vector of the view

To compute this intensity, we need to compute the direction of light reflection by using the following formula :

$$R = 2(N \cdot L)N - L \tag{10}$$

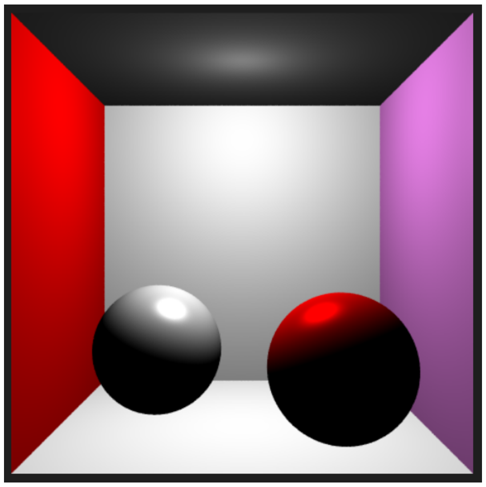


Figure 4: Result of ray tracing on Cornell's box with Phong's illumination

2 Hard/Soft Shadow

A Hard Shadow

Now we have a beautiful box with Phong's illumination, we have to add some realistic effects, such that hard shadow.

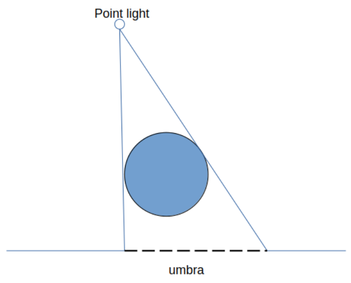


Figure 5: Scheme of hard shadow

The aim is the follow : when we throw a ray, we compute its intersection point, so, to add shadows we have to throw again a ray from the intersection point to the light source, and we do this for all the light.

After that, we have to check if the new ray intersect or not an object, and in the case where it intersects, we add a black color (*Vec3(0,0,0)*), otherwise, we do nothing.

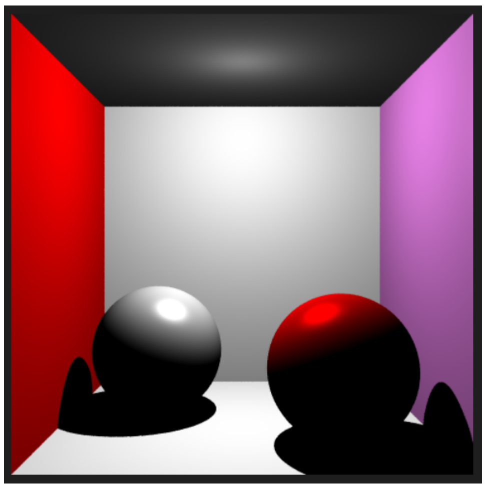


Figure 6: Result of ray tracing with hard shadows

B Soft Shadow

The soft shadow is more interesting because it's more smooth and realistic. To do soft shadows, we have to divide the light into several random point light.

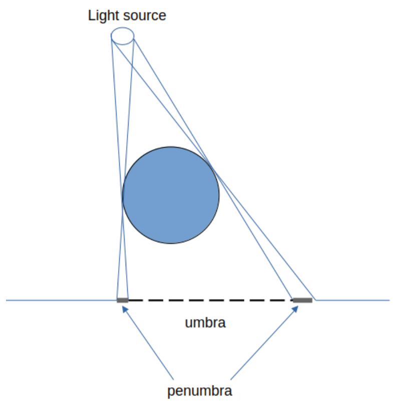


Figure 7: Scheme of soft shadow

And after that, this is the same way that hard shadow but we define a variable that we will increase when we throw the new ray and that there is not an intersection. We can modify the number of sample in order to display a render with more or less ”noise” in the shadows.

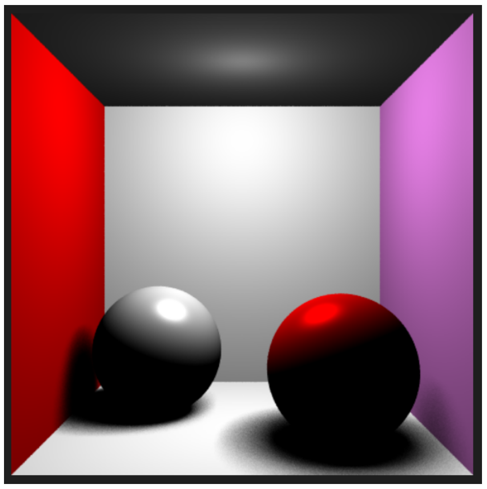


Figure 8: Result of ray tracing with soft shadows

IV STEP THREE

1 Meshes intersection

In this third part, we have to compute the intersection with a mesh. To compute the intersection point, we have to check if the ray is parallel to the triangle. To do this, we compute the dot product between the direction of the ray and the normal and check if the result is equal to zero or not. If the result is equal to zero, there is not an intersection.

Then, we have to check if the intersection point is inside the triangle. We have to compute the cross product between each edge and the vector between the intersection point and a point of the triangle, and we compute the dot product between the normal and the the result of the cross product. In each case, if the dot product is negative, there is not intersection.

And after all of this, we know if the intersection point is inside the triangle.

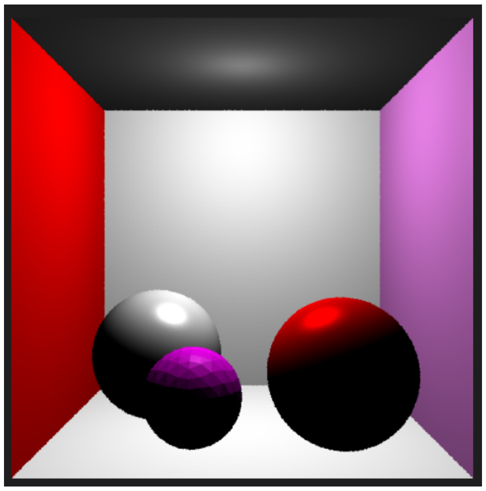


Figure 9: Result of ray tracing with a reflected sphere

V FINAL STEP

1 Reflected sphere

To compute the reflection on sphere, we use the reflection's formula in order to compute the new ray, and that recursively, i.e, we define a variable *NRemainingBounces* and we decrease it until the variable is equal to zero.

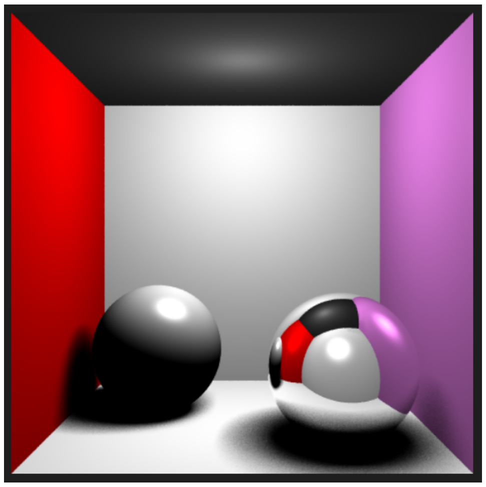


Figure 10: Result of ray tracing with a reflected sphere