Monte Carlo Ray Tracer

TNCG15 - Advanced Global Illumination and Rendering

Antonio Tällberg Ilestad (Antta839)
Tim Olsson (timol085)

October 30, 2022

ITN
Linköpings Universitet

Abstract

The goal of every rendering is to strive for photo realism. One way of achieve this is by solving the global illumination problem, which is to create realistic light and mimicking its behaviour in the scene. There is many different potential solution to implement in the renderer to get realistic lights and in this report Monte Carlo ray tracing is being used. The foundation of global illumination will be described in order to understand how the Monte carlo ray tracing works. This projects implementation is written in C++ and the created scene contains objects with different shapes and material, these objects showcases the feautures soft shadows, reflections and color bleeding. The materials consist of perfect reflectors and diffuse lambertian surfaces. In the end, a discussion covering an analysis of the results is presented with the rendering time.

Innehåll

1 Introduction

The goal of most rendering applications is to create a virtual representation of a realistic world, there it should be as realistic and have as short a rendering time as possible. There are many different methods in order to create these representations and also many shortcuts in order to shorten the time frame while still getting as good a representation as possible. The time frame often decides the method used and also which physical properties that are being used. One big decision is on the global illumination method.

The aim of this project is to learn more about the global illumination technique of Monte Carlo ray tracing and demonstrate it by implementing it in a renderer. The report covers both the basic concepts of global illumination techniques, and how it can be implemented to create a room with objects in.

1.1 Global illumination

Global illumination models strive to mimic the light in the real world such as reflected, refracted, absorbed, emitted and re-emitted. This is the difficult part because light bounces around an incomprehensible amount of times before it hits our eyes. A renderer needs to simplify this substantially and use simplification of real world physics. At this moment of time the renderers are good at global illumination methods so that we are capable of creating realistic images that are hard to distinguish from the world.

These simplifications is algorithms that is based on numerical approximations that is also called rendering equations. Many times the rendering equation 1 is used which describes the complete transport of light in a scene as emitted, reflected radiance.

$$L_s(x, \Psi_r) = L_e(x, \Psi_r) + \int_\Omega f_r(x, \Psi_i, \Psi_r) L_i(x, \Psi_i) \cos\theta_i d\omega_i \tag{1}$$

where L_s is the surface radiance in point x, L_e is the emitted radiance by the surface, which does not need to be computed as it can be found though the surface definition. The integral is describing the radiance values in the scene and can be compuded using ray tracing. L_i as the incoming radience in direction Ψ and f_r is the bidirectional reflectance distrubtion function also know as BRDF of the surface. The equation integrates over the hemisphere Ω of surface x. This is because there could be an infinite number of possible incoming directions.

1.2 Ray-tracing

By following a ray from the lightsource into the scene, ray tracing is a method for simulating the light in a scene. The path of the ray is followed through the scene as it bounces off different objects it hit its point of origin. Pixel values will be generated as the ray moves through the scene and interacts with the objects, creating a rendered image. This method is called forward ray tracing.

However, a large number of these rays will not reach the viewer thus will not have a meaningful impact on the image. To ensure that the rays are evenly dispersed equally throughout the scene, a large number of rays have to be used. Which will make the rendering process computationally expensive.

1.2.1 Whited Ray tracing

Whitted ray tracing was developed in 1980 by Turner Whitted which was one of the earlier techniques to simulate light [1]. It was based on backward ray tracing which means that the method to calculate the ray originated from the viewpoint. The big differense from other methods was that it started from the viewpoint which often is the camera instead of the lightsource. The rays get sent in the scene, see figure 1.

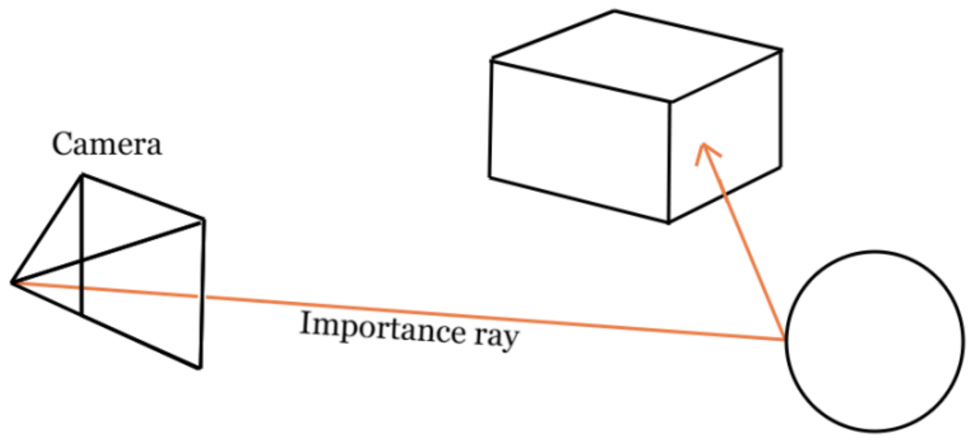


Figure 1: Ray path depending on obejcts surface normal

The basis is different becuase it can't be sent out rays and expect it to hit the lightsource, the probelility of that happening is very low. So what is done is that each ray has a predetermined amount of bounces, where at each bounce the local lightning model is used in order to calculate the contribution of each bounce position. Each ray will hit a pixel and that pixel colour will be describe as where the ray first hit.

With this it won't be neccesary to calculate all rays because only those points that will be visuable from the camera is relevant. Also the Whitted ray tracing handled bounces differently depending on the material. It depends on the reflectness and reflectionness of the material. There is three cases, perfect mirror, perfect refraction and lastly a standard absorbing material. The light travel path calulations becomes more efficient compared to the travel path in the real world and also the process becomes faster.

1.2.2 Monte carlo Ray tracing

Monte carlo ray tracing is a improve method of Whitted ray tracing where it can handle diffuse cases. The downside of the Whitted is that it only handle perfect cases with surfaces which is improved in Monte carlo method. In the real world all surfaces is not perfect mirror or reflective materials, there is also diffuse or glossy materials. A type of diffuse surface is called Lambertian which spreads the outoging light in all directions. This case is almost impossible to calculate all the ray paths so the Monte carlo method uses statistics to simulate the random directions. The main idea is to calculate one random path at the time at each intersection untill either that it hits a lightsource or other predetermined condition. Since it is possible for a ray to never reach a light source, Russian Roulette is used to terminate some of the rays inorder to reduce the amount of unneccesarly calculations.

Another benefit of Monte Carlo is soft shadowing which means that the shadows become more realistic becuase the shadows dont get as sharp edges. Instead of using one ray to check for visibility to determine the edges of an object, mulitple rays is being used so in some directions can be visible and other unvisible. That will create a more of a gradient at the edges which give an illusion of a softer shadow, see figure 2.

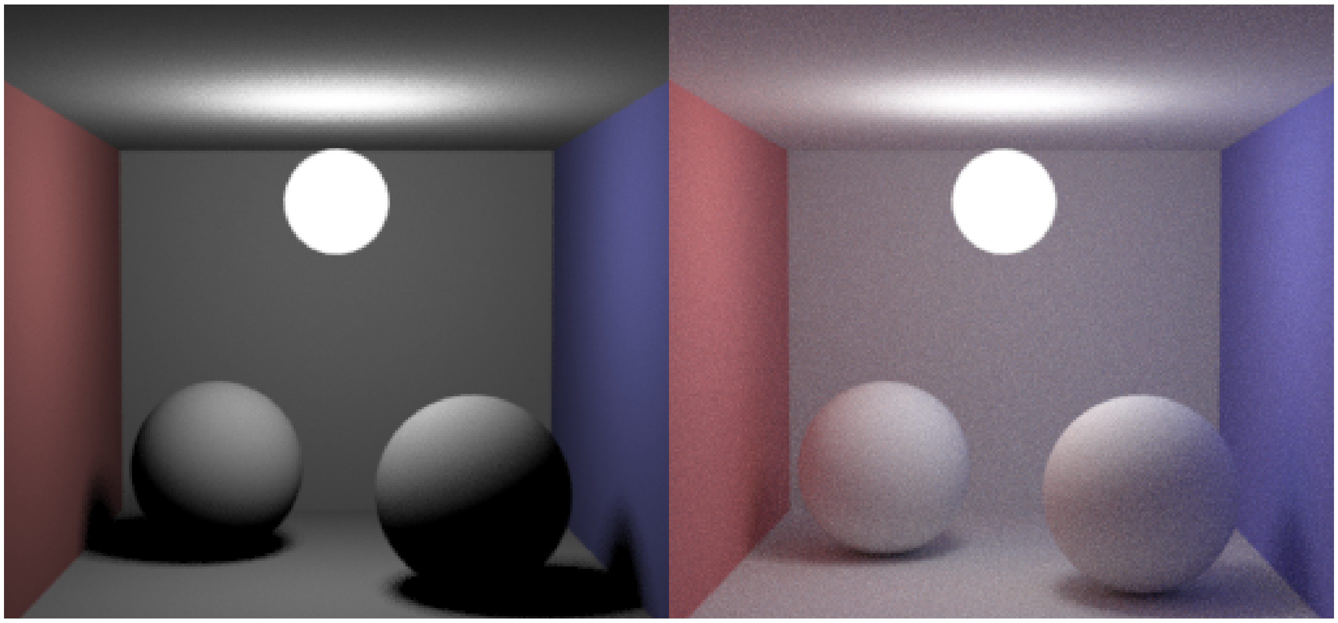


Figure 2: Left image is with hard shadows and right is with soft shadows.

2 Background

This chapter covers some basic concepts of global illumination and physics that are needed to understand and implement a Monte Carlo ray tracer. Simpler introductions to the scene, radiosity, illumination, rays and material interactions.

2.1 Scene

One of the most commonly used for test scenes in rendering is called the Cornell box which is in the shape of a cube, where the light source is positioned in the middle of the ceiling which is white. The ceiling,floor and back wall is also white, while the right and left is respectively green and red. The reason for this colour scheme of the room is that it is possible to see the colour bleeding as the white parts of the box will get a subtle shade of either red or green. With a different colour scheme on the wall, the objects in the scene with different reflection models can be tested to see if they are correctly implemented. In this project the Cornell box is not being used but a hexagonal room [2]. The scene with the hexagonal room and its description is shown in the figure is shown in the figure 3 and 4.

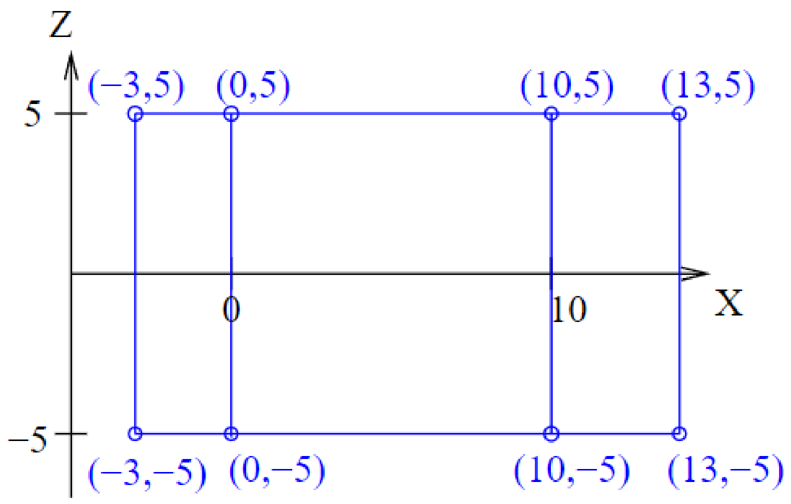


Figure 3: The room coordinates from the side

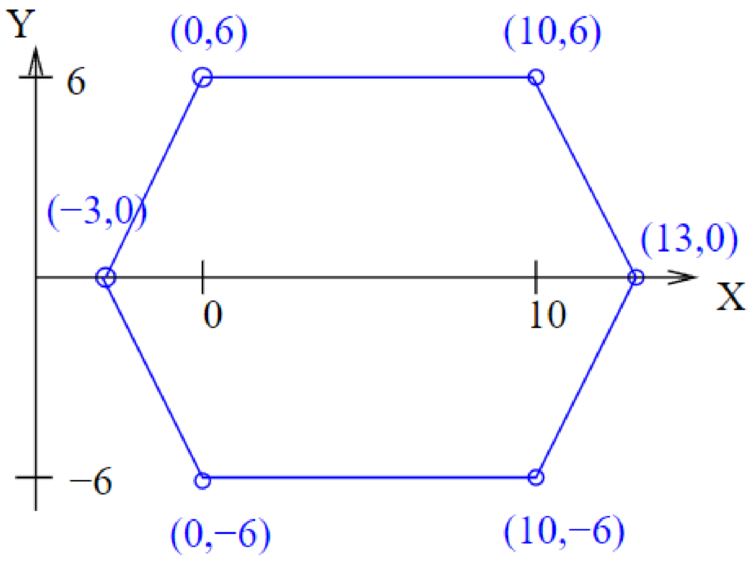


Figure 4: The room coordinates from above

The scene has the left back wall as a perfect mirror, the right back wall as green, side walls as respectively red and cyan, the floor and the celing as purple. The materials of the walls and objects in the scene that are not transparent or mirror is made of a diffuse material. The lightsource is is a sqaure lightsource with the position in the middle of the ceiling. There is also 3 objects in the scene, a two spheres and a cube. The cube is made of a normal diffuse material.

2.2 Ray

The first step was to define a ray, it was constructed with a starting point, an endpoint and a normalized direction.Each ray that is sent out has importance which describes how important that ray is for the rendering. The importance for the ray always start with a value of one and for each bounce and the depth of the ray decreases that value. The importance is being used in the calcutation of the radiance everytime a ray intersects with a surface. For simplicity the ray tracer always assumes that the radiance is the same in the begining of a ray as its end point. The radiance value consist of a colour as well as the intensity.

As the ray hit a diffuse surface, the endpoint was set to the intersection point between the ray and the surface. The colour of the ray is set to the same colour as the surface that it collided with. This will in turn cause the effect that was mentioned in the scene section, colour bleeding. Because when the ray hit the its next surface it will have the colour from the previous surface. This will continue until the ray is terminated.

When undersampling unwanted patterns can appear, these patterns are called aliasing. This can be solved by sending more rays into one pixel to make the necessary calculations to get the pixels average value. The more rays that are sent, the more intersections points can be used to get the average value of that pixel.

2.3 Ray intersection

To determine if a ray hit a surface plane within the scene a intersection algorithm was implemented, the Möller-Trombore intersection algorithm. Two different types of planes can be found in the scene, rectangles for building the room and triangles for the objects in the room. The scene also contains spheres. Therefore, there will be three different intersections algorithms needed for the scene.

2.4 Illumination

For a realistic scene to be created, the distribution of light were calculated, both direct and indirect illumination.

2.4.1 Direct illumination

To calculate the direct illumination on objects in the scene shadow rays were created. The shadow rays are also used for calculating the shadows in the scene. A shadow ray was created at the intersection point between a surface and a ray that was cast from the eye, from this intersection it was cast toward the light source. By checking if the ray hit any object in between the eye on its way to the lightsource it can be calculated if the object lies in shadow or are directly illuminated. The more shadow rays that are used the more realistic the rendered image will become, it will results in a softer shadow since there would be more samples. This will also increase the rendering time.

2.4.2 Indirect illumination

Indirect illumination is used for realistically create and visualize the scattering of lights. To compute the indirect illumination the rays that intersected with diffuse surfaces were redirected in a random direction. At each bounce the radiance of the ray was reduced to get a realistic amount of light left in the scene. Until the ray is absorbed by a surface these steps will happen recursively. To determine if the ray is absorbed or not, russian roulette is used were a certain percentage of the rays are terminated at random.

2.5 Radiosity

Radiosity is a global illumination technique for lighting a scene with indirect illumination. Areas which are not under direct illumination is still observable and visiable. This is becuase ray bounces from other surfaces so it can be describe as calculating the diffuse refelction in a scene. In order to lessen the burden of the calculation from common ray tracing method, radiosity is used to calculate this diffuse light. So the scene gets a more relasitc imagery without needing to use a unrealistic ambient illumination. It is based on every surface is divided into smaller surface pathes which will get an intensity value dependent on the lightsources and the other patches. The lighting leaving a poatch is than a combination of the light being emitted and reflected by the surface. The intensity values are independing of the veiwing direction which means that the values can be pre-computed and used in real-times applications.

2.6 Bi-directional Reflectance Distribution Functions

In the physical world, energy never disappears and only is transformed into new forms. Light which is photons hits an objects surface will get bounced off but depending on the wavelength some of the photons will be absorbed and transform to heat. These photons will always change keep moving which is the part of absorbed and re-emitted. This event is simplified with our BRDFs for our renderer depending on the material properties that is being handled. Some of the usual materials that is handled is mirrors, transparent and lambertian reflectors.

2.6.1 Reflective material

Reflection can be divided in two parts, ideal or perfect reflection and diffuse reflection. In the ideal reflection the energy is preserved and that the outgoing is the same is the ingoing in relation to the surface normal, see figure 5 [3]. This creates a mirror effect for the surface becomes no rays is absorbed and only is reflected.

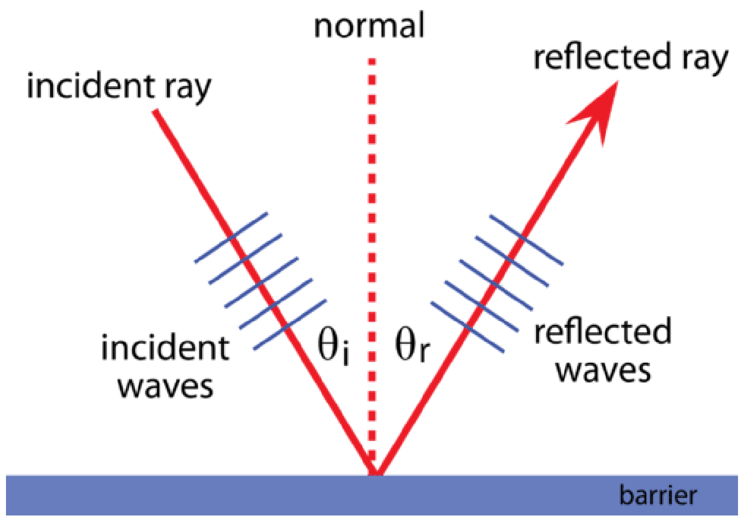


Figure 5: How light get reflected on a perfect surface

For diffuse reflection is the case that the surface is not perfectly smooth which means that the outgoing and ingoing does not have the same angle in relation to the surface normal, see figure 6.

The rays will be able to reflect in all direction and also some of the wavelength can be absorbed of the light, this creates the bleed effect on other close objects.

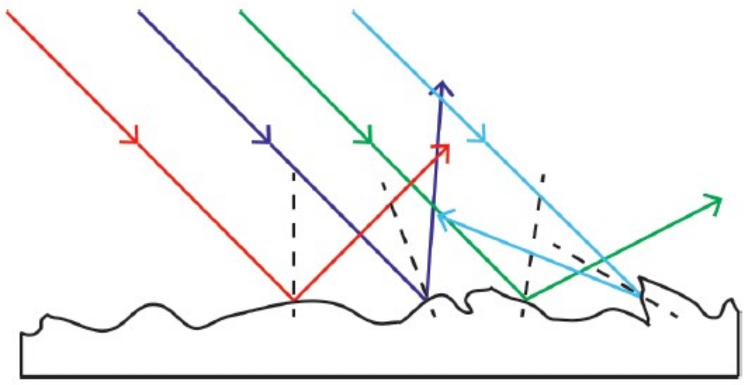


Figure 6: How light get reflected on a uneven surface

2.6.2 Refracting materials

Refraction which occurs when a light moves from one medium to another and it will bend the light to another direction depending on the new medium, See figure 7[4]. These materials is transparent material which often handles medium as air,glass and water. The light will follow Snell's law, which bases the refracted light depending on the refractive index between medium before and after intersection. Depending on the incoming lights angle towards the objects surface normal, parts of the light might get reflected or refracted.

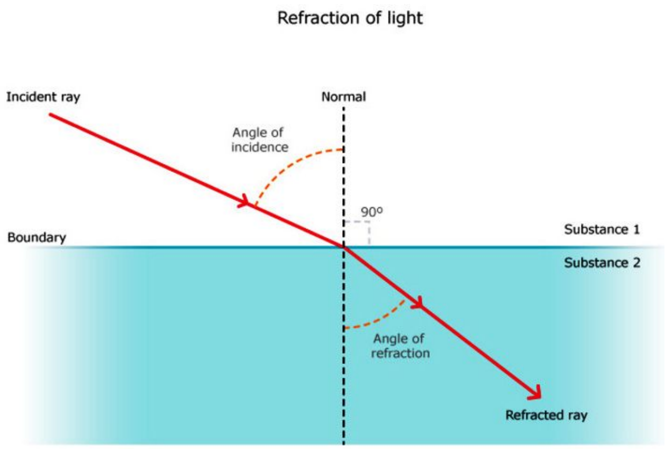


Figure 7: How light get refracted

2.6.3 Lambertian reflector

Lambertian reflectors is surfaces that has the properties of having the radiance getting emitted in all direction, so in these cases BRDF becomes constant, see equation 2. ρ is called the reflectivity of the surface, the division by π is for energy conservation.

$$f_r(x,(x,\omega_{in},\omega_{out}) = \frac{\rho}{\pi} \tag{2}$$

The problem with this is in order to calculate this, it would need a infinite computing power in order to follow every possible ray. So probability theory to calculate this and is where Monte carlo differentiate from Whitted ray tracing. So instead of following all the rays, we choose random rays to be drawn from a probability distribution. The probability distribution function is defined as $p(x)$ which mathematically describes the probability for a random variable. The function needs to fulfill two conditions, see equation 3 and 4.

$$p(x) \geq 0 \tag{3}$$

$$\int_{-\infty}^{\infty} p(x)\,dx \tag{4}$$

The PDF can then be used in relation with a cumulative distrubtion function (CDF). CDF $P(x)$ is a cheaper and faster version which will take the probability that a random variable is less than x or equal [4], see equation 5.

$$P(x) = \int_{-\infty}^{x} p(x)\,dx \tag{5}$$

The CDF draw random number according from a PDF see equation 6.

$$x_i = CDF^{-1}(y_i) \tag{6}$$

y_i is the random number from the unifrom PDF and x_i is the number that follos our PDF. The invese CDF will always be defined since the CDF is stricly increasing.

3 Results

Now the theory is done for the global illumination and rendering, in this section it is shown how changing the parameters will affect the image. The usual parameters that have a big impact on the end result is rays per pixel and the amount of shadow rays.

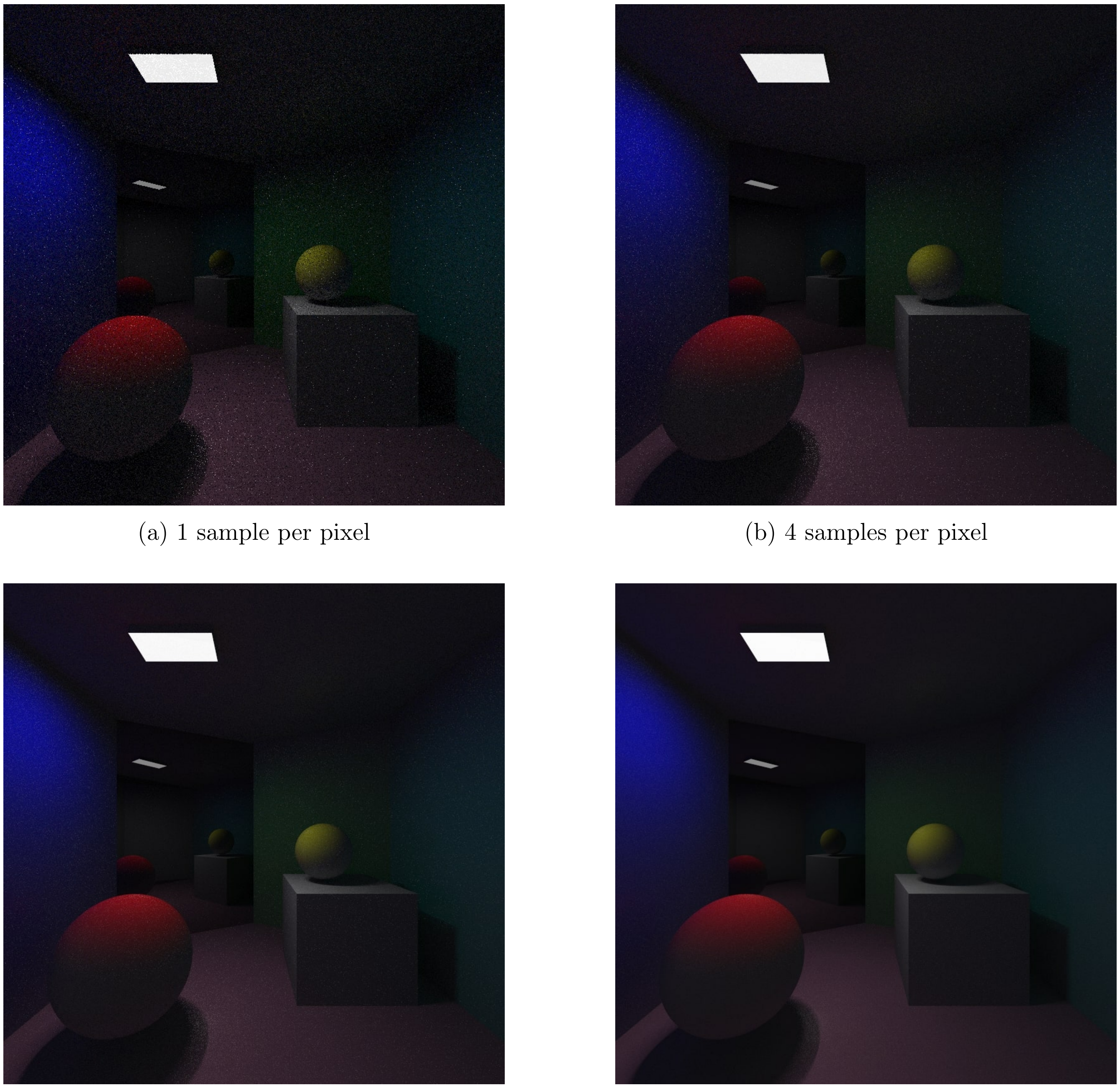
This was done by implementing a Monte carlo tracing method in C++ with the OpenGl Match library(GLM). There GLM helps with creating vectors and the mathematical equations that is used. The size of the rendered images is 800x800 pixels. The scene which were described ealier is a hexagonal room with two spheres in different colours and one cube, there is one wall that is a perfect reflective mirror.

3.1 Rays per pixel

Rays per pixel, also known as samples per pixel is the amount of rays for each pixel. So the higher count the more information is sent to the renderer which will give a higher detailed image. The cost will also vary, the higher amount of rays, the higher computing power is needed, as can be seen in table 3.1. Below in figure 8 are the resulting images from the different settings that are shown in the table.

Table 1: Relationship between rendering time and sample per pixel

Image	Rays per pixel	Rendering time
1	1	5 s
2	4	51 s
3	16	1min 32 s
4	64	5min 50 s



(a) 1 sample per pixel

(b) 4 samples per pixel

(c) 16 samples per pixel

(d) 64 samples per pixel

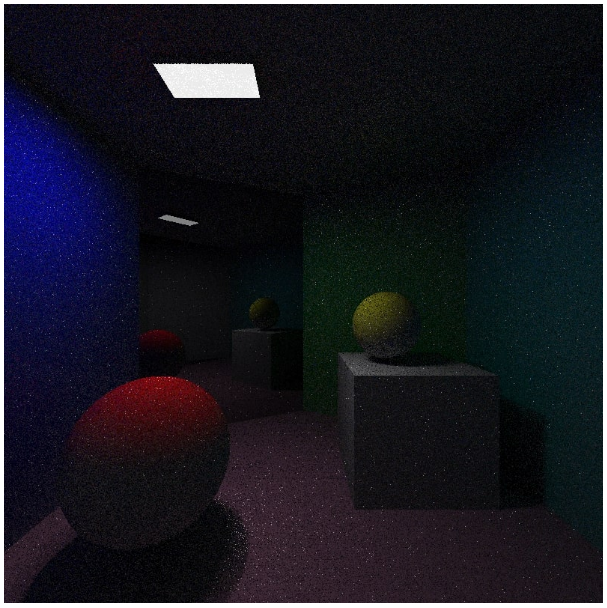
Figure 8: The rendered image with a different number of samples per pixel

3.2 Shadow ray

In table 3.2 the amount of shadow rays in comparison to its rendering time is shown. Here it is clearly shown that the increase of shadow rays also increases the cost of rendering time. In figures 9,10 the resulting images for the diffent number of shadow rays are shown.

Table 2: Relationship between rendering time and shadow ray count

Image	Shadow ray count	Rendering time
1	1	5 s
2	2	9 s
3	4	15 s
4	8	28 s


(a) 1 shadow rays

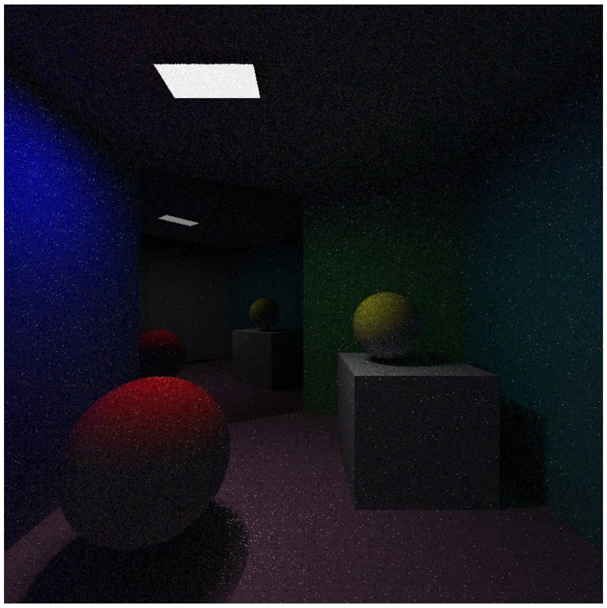
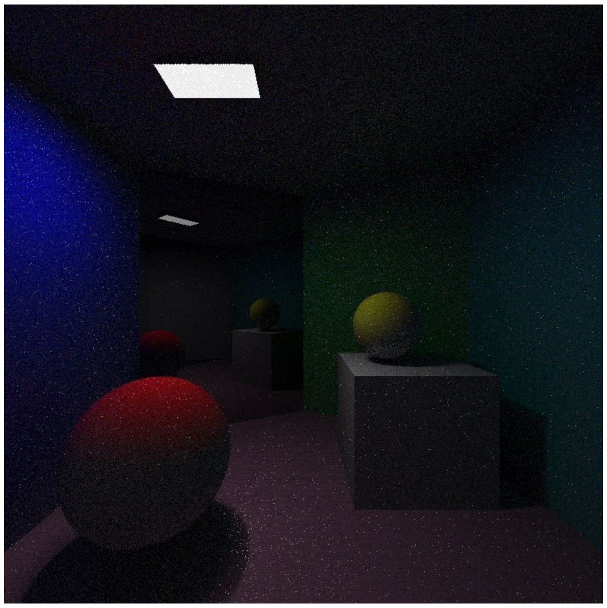

(b) 2 shadow rays

Figure 9: The rendered image with a different number of shadow rays


(a) 4 shadow rays

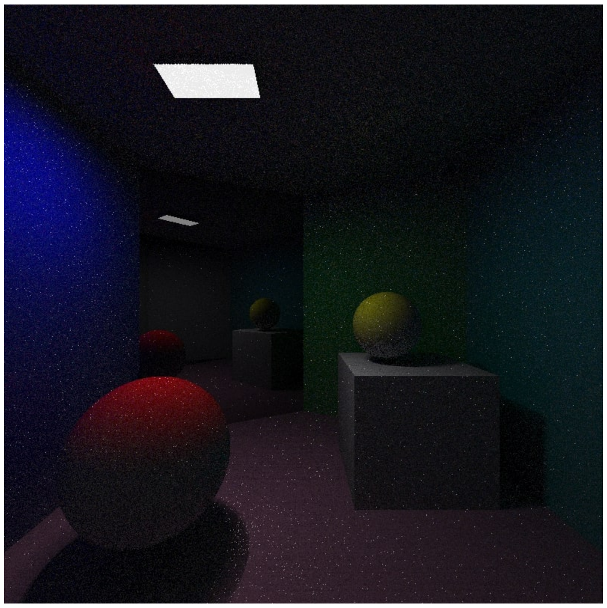

(b) 8 shadow rays

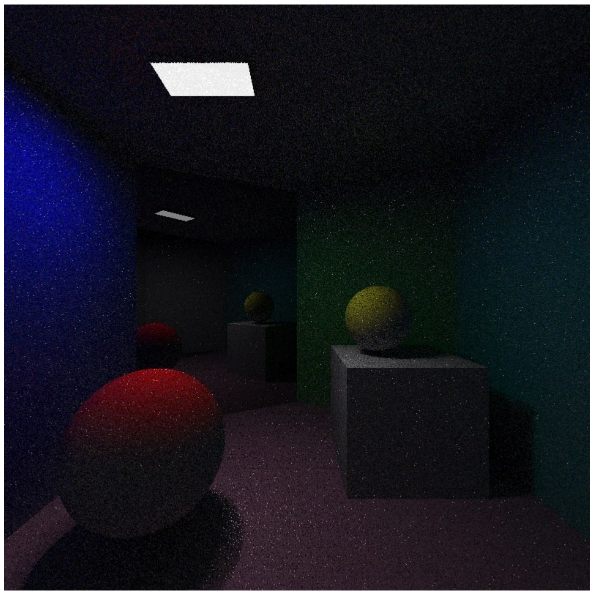
Figure 10: The rendered image with a different number of shadow rays

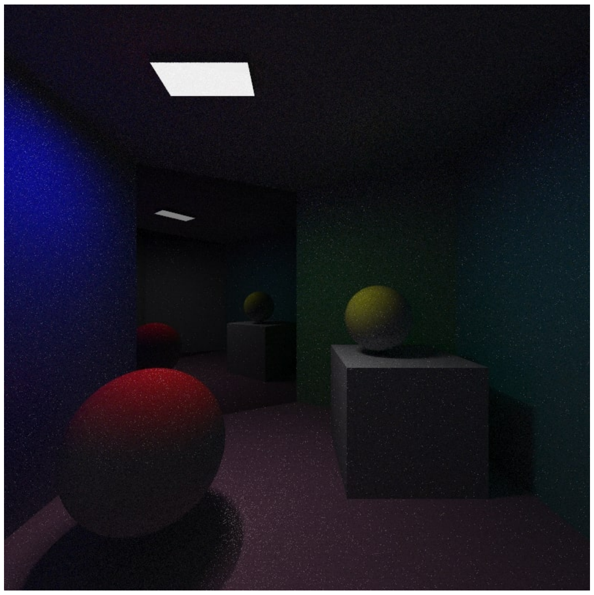
3.3 Combining parameters

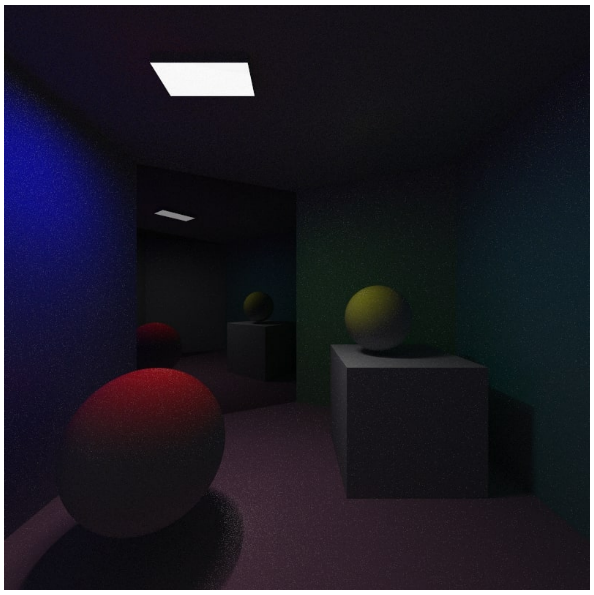
Here it can be seen how the image improves more significantly when both the shadow ray amount and sample rays is increases. The computing power needed is higher but the change is clear even in the smaller details, see table 3.3 and figure 11.

Table 3: Relationship between rendering time and different parameters

Image	Ray per pixel	Shadow ray count	Rendering time
1	1	1	5 s
2	4	2	35 s
3	16	4	4 min 10 s
4	64	8	29 min 55 s


(a) 1 sample and 1 shadow rays


(b) 4 samples and 2 shadow rays


(c) 16 samples and 4 shadow rays

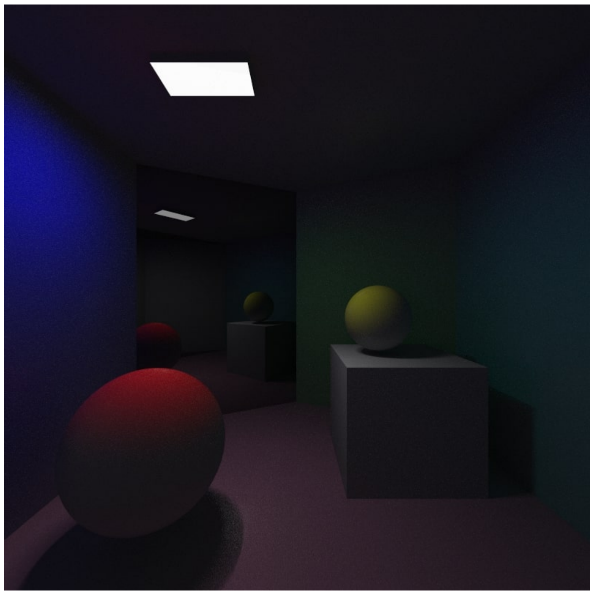

(d) 64 samples and 8 shadow rays

Figure 11: The rendered image with a different number of samples and shadow rays

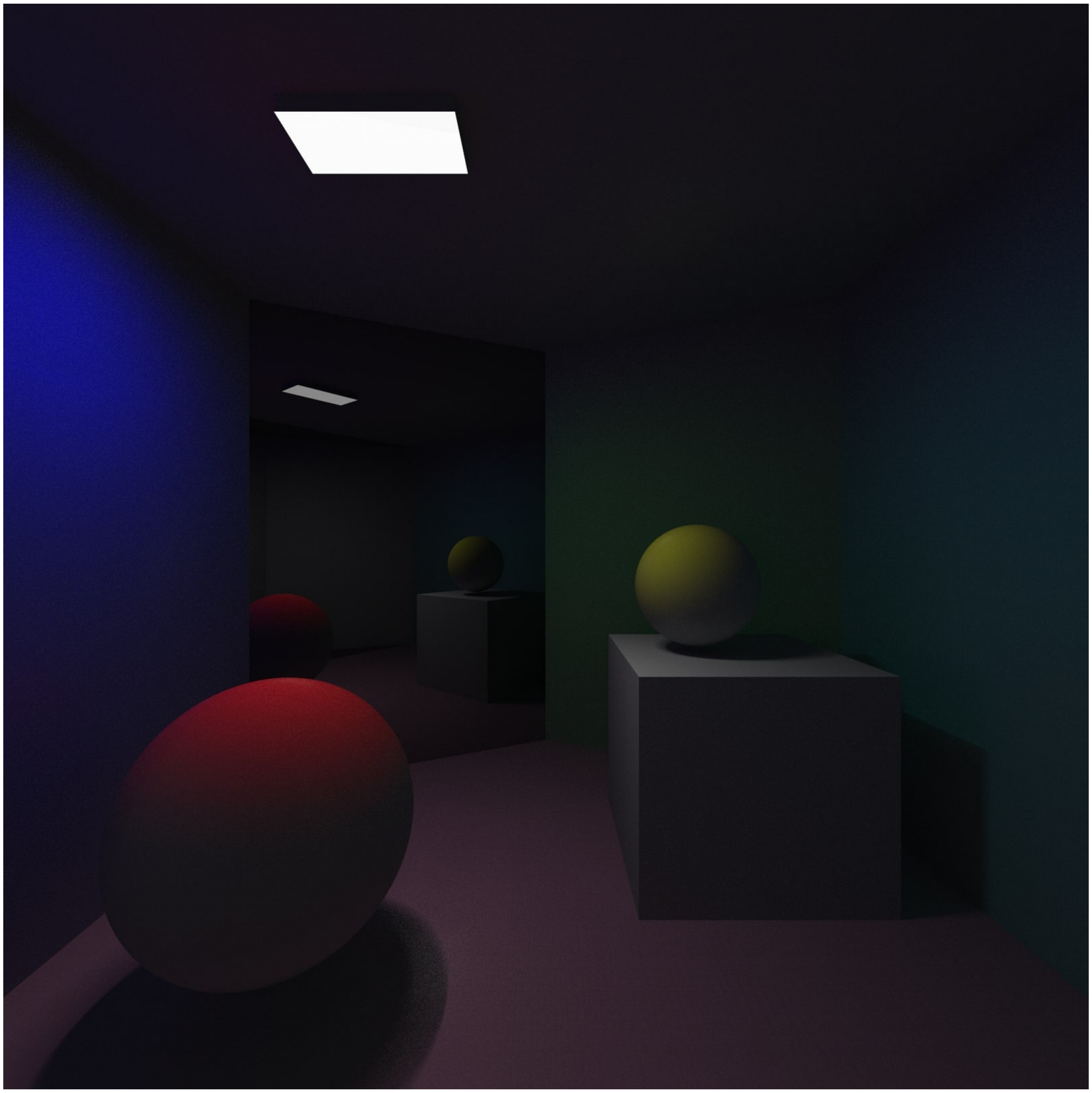


Figure 12: 64 samples, 8 shadow rays and a resolution of 1600x1600

As an additional test the resolution of the image was also increased as well as the two pervious parameters. This resulted in a quite realistic image as seen i figure 12, this however also resulted in a large rendering time of 2 hours and 25 seconds.

4 Discussion

As seen in the result section, increasing the number of samples per pixel and shadow rays resulted in a more realstic image. When the samples per pixel got increased it produced a image with less noise compared to a lower amount of samples, this is best shown comparing 8a with one sample compared to 8d with 64 samples. When comparing the rendering time for the image with 64 samples with the one with one sample, the rendering time had gone up drastically with roughly 7000% (Table 3.1). The image with 16 samples per pixel produced overall the best image quality to rendering time ratio.

When comparing the different results for the shadow rays the change was not as drastic as when increasing samples. The shadows did turn out better and with a softer edge instead of a hard one but that was only when zooming in to the image. This is not something that draw the same attention to it at first sight as the nosie does for a low sample rate. It can be argued that a scene with a different lighting setup and objects could show a more drastic difference, but from this scene the difference is not that noticeable at first sight. From this the conclusion could be made that only increasing the shadow rays does not result in a more drastically realstic image.

The Monte Carlo ray tracing technique is a relatively straightforward for modeling scenes, but it has also requires large computational power. This implementation of Monte Carlo is slow and noisy, as is typical of the algorithm. To reduce the noise level more samples were used but this resulted in a lot longer rendering time. A image with a high amount of samples, shadow rays and increased resolution was created in the end. This image undoubtedly looks the best, but when the rendering time is over two hours its not really a viable option with this method.

The final results from the Monte Carlo ray tracer gives a decenlty good visualization, with the downside of long rendering time. These are problems that could be fixed with a more efficent structure of the code or with brute force from a more powerful computer. Or with other options as hardware accelerated rendering or multithreading.

References

[1] J. D. Foley and T. Whitted, *An improved illumination model for shaded display*, 1980.

[2] Mark E Dieckmann. *"TNCG15 Advanced Global Illumination and Rendering: Lecture 4 ("The scene 1")* . MIT group, ITN, 2022.

[3] D. Halliday, R. Resnick & J. Walker *Principles of Physics*, Eleventh edition. John Wiley & Sons Inc, 2014

[4] Mark E Dieckmann. *"TNCG15 Advanced Global Illumination and Rendering: Lecture 8 (Idea of MC scheme),* . MIT group, ITN, 2022.