Monte Carlo Ray Tracer

TNCG15 - Global Illumination and Rendering

Tilda Hylander <tilhy349@student.liu.se>
Jonathan Andersson <jonan270@student.liu.se>

November 13, 2021

Abstract

A Monte Carlo ray tracer is a type of rendering method which uses ray tracing to produce photo-realistic images. It makes use of global illumination to produce realistic effects such as color bleeding which local lighting models struggles to produce. This report aims to implement such a rendering method and evaluate the results. The resulting implementation of the ray tracer is used to render a pre-defined scene which contains renderable objects in the form of spheres, triangles and tetrahedrons. The objects are given material properties via a *BRDF*. Implemented material properties include lambertian reflectors and perfect mirrors.

An area light source based on a surface consisting of triangles is included in the scene to generate realistic soft shadows.

The resulting work is a working Monte Carlo ray tracer. The rendering of the scene is presented for a varying number of samples. By looking at the resulting images it is clear that the image quality is directly related to the number of samples used. The results are discussed and improvement points are presented.

Contents

List of Figures

1 Introduction

Rendering photo-realistic scenes has long been a popular subject in computer graphics. There are a number of rendering methods available where some are tuned towards efficiency while others are aimed at high levels of photorealism. Rasterization is the fastest way to render predefined scenes, but photorealism is lacking in points of reflection or refraction, which is where ray tracing methods excel [1]. For some types of scenes, realistic reflection and refraction is required for a sense of adequate photorealism. This can for example be in areas such as optical system design, lighting fixtures or lamp design [2].

Ray tracing is a method of graphics rendering that simulates the physical behavior of light. A ray tracer uses simple algorithms to produce realistic images. The popularity of ray tracing has increased due to increases in computing power and also because of its ability to cleanly handle effects such as shadows and transparency [3].

1.1 Global Illumination

Global Illumination is the more advanced form of ray tracing which add to the local illumination model by reflecting light from surrounding surfaces to the object. A global illumination model is described to be more comprehensive, more physically correct and also produces more realistic images [4].

Compared to other rendering methods, global illumination models have a more realistic way of modeling light. The method presented in this report uses a global illumination model which utilizes ray tracing in order to achieve a high level of photorealism.

1.2 Required steps for Global Illumination

Global illumination aims to simulate both direct illumination and indirect illumination which in combination creates a realistic image. Light can be contributed to a point by either a light source or by a reflection via another surface in the scene. Global illumination takes this into account when calculating the color of a point in a scene.

Light is assumed to be transported using rays. A light ray is defined by its starting point and end point and carries radiometric values.

In order to be classified as a global illumination model, three steps need to be specified. The first step is to define the area in which the global illumination model is applied, which we will refer to as the *scene*. The scene contains all 3D objects that is to be rendered aswell as their material properties. The scene also contains light sources which interact with the objects within the scene. The second required step is to use the predefined scene to obtain radiometric values of light. These are then converted into pixel color values in the third and last required step[1].

1.3 Radiometric values to represent light

In order to apply a global illumination model, some kind of value that represents color and light is required. This value is transported along rays within the scene and modified according to intersection points. This radiometric value is called *radiance* and represents light leaving a certain point, in a certain direction. These radiometric values are then converted into pixel colors by some predefined way, as required by a global illumination method.

Another way to interpret the relationship between incoming and outgoing radiometric values at a point is to look at the ratio between incoming and outgoing light from the opposite direction. This value is known is *importance* and is traced by emitting rays from the eyepoint[2]. The ratio between incoming and outgoing radiance is equal to the ratio of incoming and outgoing importance[3].

1.4 Material description

Photorealistic representations of material properties can be a difficult subject. Many real world materials have complex properties with regards to specular reflections, opacity reflectivity and emission. Material properties can be described by a bi-directional, reflectivity and distribution function (BRDF) [5]. The rendering method described in this report have two different types of materials: perfect mirrors and Lambertian reflectors.

[1] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 1 (Introduction), 2021
[2] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 4 ("Whitted ray-tracing 1"), 2021
[3] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 2 (Rendering equation), 2021

1.5 Whitted Ray Tracing

An obvious way to implement a ray tracing algorithm would be to trace light from the source and follow the path until the viewer is reached. Another approach is following the path in the opposite direction, from the viewer into the scene which is not as wasteful since it eliminates the risk of rays never arriving at the viewpoint [5]. This type of ray tracing is applied in the so called Whitted ray tracer method as described by Turner Whitted.

The Whitted ray tracer uses a global approach to render realistic scenes. The initial rays are emitted from the camera and followed through the scene. The rays will intersect with objects and spawn new reflective rays, thus creating a tree structure of rays. When a ray hits an object in the scene, it is given the same angle of reflection as the inclination angle against the surface normal. The Whitted ray tracer also simulates specular reflections by adding random perturbations to the normals of intersected objects. The variance will depend on material properties where a higher variance results in a glossier appearance [5].

The Whitted ray tracer method is however not ideal for rendering diffuse surfaces. Instead of containing components due to reflections of nearby objects and light sources, diffuse reflection is instead represented by a diffuse term similar to the Phong shading model. [5].

1.6 Project implementation

This project aims to implement a Monte Carlo ray tracing method, which is in many ways based on the Whitted ray tracer. The Monte Carlo ray tracing method approximates radiometric values by relying on a combination of sampling and applying noise to reduce aliasing [2]. The Monte Carlo ray tracing method also differs from Whitted ray tracing for calculating reflections. The direction of reflections for Monte Carlo ray tracing are not given the same angle of reflection as the inclination angle for diffuse surfaces.

In order to implement the model, the programming language *c++* was used. The system architecture relies heavily on object orientation to make the code easily readable.

2 Theory and method

The idea of ray tracing is to compute the *irrandiance* arriving at each pixel in the *image plane*. The irrandiance arriving at a point is also the sum of the radiance falling onto that point from all directions[1]. A global illumination model will be applied to figure out how much light is falling on a certain point and what color will be produced.

2.1 Scene Description

The scene that is constructed and used to display the results is approximately illustrated in Figure 2.1. It is a closed room that consists of six walls in the shape of a hexagon, a roof and floor, all which has a reflectance coefficient of $\sigma = 0.7$. The front right wall acts as a perfect mirror. The camera plane is square with the side length 2 located with its center at the origin (0, 0, 0). The room also contains a perfect mirror sphere with radius 2, a lambertian sphere with radius 0.7 and $\sigma = 0.2$ and also a lambertian tetrahedron with side length 4 and $\sigma = 0.6$.

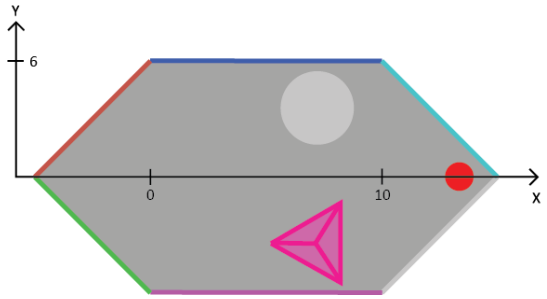


Figure 2.1: Approximate illustration of the scene

2.2 Rendering Equation

A model which aims to describe the radiance emitted from a point x into a direction ω_{out} is the rendering equation, see equation 2.1 [4]. $L(x \rightarrow \omega_{out})$ is the radiance reflected

[4] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 3 ("Radiosity"), 2021

from the point x in the direction ω_{out}, $L_e(x \to \omega_{out})$ is the emitted radiance at the point x (the direct illumination) in the direction ω_{out} and the integral is a sum of all indirect light coming from all directions ω.

$$L(x \to \omega_{out}) = L_e(x \to \omega_{out}) + \int_{\omega} f_r(x, \omega_{in}, \omega_{out}) * L(x \leftarrow \omega_{in}) cos\theta_{in} d\omega_{in} \tag{2.1}$$

2.2.1 Monte Carlo Integration

A numerical method for solving the rendering equation is Monte Carlo Integration. The general idea is to solve the integral part of 2.1 using Monte Carlo Integration. The integral cannot be solved analytically and therefore it must be approximated in some way. In Monte Carlo integration, random numbers are used to approximate integrals. The expected value of the integral can be estimated using a sum of finite samples N. The estimator for the radiance $L(x \to \omega_{out})$ for a reflection at a point x on a lambertian surface with reflection coefficient σ is defined as equation 2.2 [5].

$$\langle I \rangle = \frac{\pi\sigma}{N} \sum_{i=0}^{N-1} L(x \leftarrow \omega_{in,i}) cos\theta_{in,i} sin\theta_{in,i} \tag{2.2}$$

2.3 Ray-Surface Intersections

An emitted ray will intersect objects in the scene and reflected rays will spawn. In the scene there are two types of objects, spheres and combinations of triangle meshes. An intersection of a ray can be described with a value t in $x(t) = p_s + t(p_e - p_s)$ where p_s and p_e is the start- and end point respectively of the ray. The first intersection of a ray is the intersection with the smallest value of t. Intersections between rays and these objects are determined in different ways depending on the intersected object.

2.3.1 Sphere Intersections

A point with the coordinate vector x is located on the surface of a sphere with the radius r and the center c if the following equation 2.3 is satisfied .

$$||x - n||^2 = r^2 \tag{2.3}$$

[5] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 9 ("Rendering equation"), 2021

A point x on a ray with the starting point p_e and the normalized direction l is given by $x(t) = p_e + tl$. Each value t corresponds to one point x on the ray. Rewriting 2.3 with the parametrization of a ray with $a = l \cdot l$, $b = 2l \cdot (p_e - c_p)$ and $c = (p_e - c_p) \cdot (p_e - c_p)$ gives equation 2.4, where c_p is the center point of the sphere[6].

$$t = -\frac{b}{2} + \sqrt{\frac{b}{2}^2 - ac} \tag{2.4}$$

The equation corresponds to the two intersection points t_0 and t_1 between the ray and the sphere. The condition $t_0 > 0$ and $t_1 > 0$ must be satisfied since a negative value of t corresponds to an intersection point behind the ray origin. The smallest of the two values of t is chosen as the first intersection point.

2.3.2 Triangle Mesh Intersections

The algorithm used for computing ray intersections with triangle meshes is the *Möller Trumbore algorithm*[6]. A triangle is defined by the corner points v_0, v_1 och v_2. With barycentric coordinates (u, v) a point in the triangle is given by $(u, v) = (1 - u - v)v_0 + u*v_1+v*v_2$. (u, v) is defined with the conditions $u \geq 0$, $v \geq 0$ and $u+v \leq 1$. The Möller Trumbore algorithm gives the value of t, u and v in equation 2.5 using: $T = p_s - v_0$, $E_1 = v_1 - v_0$, $E_2 = v_2 - v_0$, $P = l \times E_2$ and $Q = T \times E_1$.

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{P \cdot E_1} \begin{pmatrix} Q \cdot E_2 \\ P \cdot T \\ Q \cdot l \end{pmatrix} \tag{2.5}$$

For objects which contain multiple triangles, the algorithm is performed once for each triangle.

2.4 Surface properties

Two different types of surfaces are modelled in our scene, perfect mirrors and lambertian surfaces. The reflection of an incoming ray is determined by the material.

2.4.1 Perfect mirrors

Perfect mirrors act like perfect reflectors. Perfect reflection can be modelled using Whitted ray tracing. A ray of light l is launched at a light source, either directly from a light

[6] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 6 ("The scene"), 2021

source in the scene or via a reflection. At the ray-surface intersection, a normal N is defined. The perfect reflection law states that the reflected ray r should have an angle between itself and the normal equal to the angle α between the incoming ray and the normal, see Figure 2.2[6].

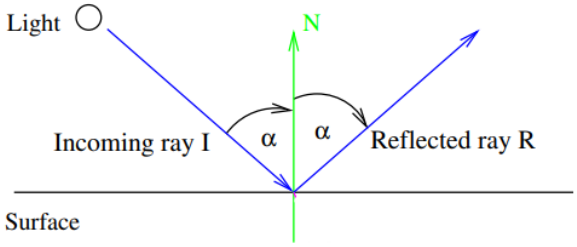


Figure 2.2: Model for perfect reflection.

2.4.2 Lambertian surfaces

Lambertian surfaces act differently compared to perfect reflectors. The reflection of an incoming ray with the direction ω_{in} that hits a lambertian surface is reflected in a random direction. The direction is best computed in a local coordinate system with the intersection point as origin. The direction of the reflective ray ω_{out} depends on the random inclination angle θ_{out} and the azimuth angle ρ_{out}. The end point of the reflected ray is $x = sin(\theta_{out}) * cos(\rho_{out})$, $y = sin(\theta_{out}) * sin(\rho_{out})$ and $z = cos(\theta_{out})^5$. Diffuse reflectors can change the color of the reflected light and this is handled by multiplying the reflectance coeffient σ with the color of the surface.

2.5 Light contributions

Light is contributed to a point by direct light and indirect light.

2.5.1 Direct light illumination

Direct illumination is computed only for those point along the ray path or at the end of the ray path which are located on Lambertian reflectors. In the project one type of light source was implemented, an area light source.

Area Light Source

An area light source is defined as a square light with a specified side length and consists of two triangles. A point is illuminated if the light source is visible to the point. The emitted light $L_e(x \rightarrow \omega_{out})$ from the point x on the surface in the direction ω_{out} is defined by the following integral in equation 2.6 which goes over all directions A over the hemisphere. σ is the reflectance coefficient, $r(\omega_{in})$ returns the end point of the ray from x along ω_{in}, $L_D(x \leftarrow \omega_{in})$ is the direct light contribution to the point x from the direction ω_{in} , $V(x, r(\omega_{in}))$ is the visibility factor and $G(x, r(\omega_{in}))$ is the geometric factor.

$$L_e(x \rightarrow \omega_{out}) = \frac{\sigma}{\pi} \int_A L_D(x \leftarrow r(\omega_{in})) * V(x, r(\omega_{in}) * G(x, r(\omega_{in}))dA \tag{2.6}$$

The integral can be solved numerically using the estimator in equation 2.7 [7]. The estimator uses N random points $x_{L,i}$ on the light source to evaluate the integral. The random points on an area light with side length s are defined using the parametrization (c_1, c_2) where a point on the area light is described with $x_L(c_1, c_2) = v_0 + c_1 e_1 + c_2 e_2$ where $0 \leq c_1, c_2 \leq s$ and v_0, v_1, v_2 are corner point of the light.

$$\langle I \rangle = \frac{\sigma L_0}{\pi N} \sum_{i=0}^{N-1} \frac{V(x, x_{L,i}) G(x, x_{L,i})}{1/s^2} \tag{2.7}$$

The geometric factor $G(x, x_{L,i})$ is defined in equation 2.8. S is the shadow ray from x to x_L, $cos\theta_L = S \cdot N_x / ||S||$ with N_x being the normal of the surface at point x and $cos\theta_L =_S \cdot N_L / ||S||$ with N_L being the normal of the area light.

$$G(x, x_{L,i}) = \frac{cos\theta_{in} cos\theta_L}{S^2} \tag{2.8}$$

Visibility is determined by sending a *shadow ray* to a point on the light. A light source is considered as visible if the closest intersection of the shadow ray is with one of its two triangles. The contribution of light that an area light has on a point is computed by sending multiple shadow rays to random points on the area light.

In order to get a visualization of the area light in the scene, the same triangle meshes which the light is made up of were added to the scene as renderable objects. If rays emitted directly from the camera hit the light source the ray is terminated and the radiance is set to a bright white value.

[7] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 10 (Light source and raytermination), 2021

2.5.2 Indirect light illumination

Illumination by indirect light means that surrounding objects reflect part of their direct light onto surrounding objects, which may or may not be directly illuminated. This is represented by the integral part of 2.1.

Indirect lighting is important since scenes rendered without it can look harsh and artificial. It is also a computationally heavy phenomenon to model [2].

Indirect illumination results in color bleeding. This happens when color is transferred between objects due to diffuse surfaces being illuminated by indirect light [1].

The calculation of indirect light contribution is made by constructing a ray tree of the intersections between a ray and object surfaces in the scene.

2.6 Ray trees

The idea of global illumination is that light can be contributed to a point by direct light from a light source and also by the reflection of light from other surfaces. This can be accomplished by creating ray trees. A ray tree consists of intersection nodes where an incoming ray hits a surface and reflected rays are spawned.

The goal is to compute the radiance that is emitted from a point x towards the eye, but as described in 1.5, generating a tree by emitting rays from light sources to obtain radiance values is not computationally efficient. Instead, rays are emitted from the eye into the scene and the nodes of the tree are given values based on the relationship of incoming and outgoing importance instead. Once a ray is terminated, only direct light contributes to the radiosity value. The tree can then be traversed backwards to obtain direct and indirect illumination for the other intersection points[8].

2.6.1 Russian Roulette

Russian roulette is the name used to describe a method for terminating rays emitted into the scene[7]. In order to simulate glossieness in materials, we can connect a variable σ to the BRDF of objects in the scene and let this variable affect the probability of ray termination. The desired effect is to have glossy objects receive more indirect light in order to simulate different levels of reflectivity.

A way to implement this effect is to randomly generate angles $\rho \in [0.01, 2\pi]$ and $\theta \in$

[8] Mark Eric Dieckmann TNCG15: Advanced Global Illumination and Rendering, Lecture 5 ("Importance and Whitted ray-tracing 2"), 2021

$\left[0.01, \frac{pi}{2}\right]$ at an intersection point, and use that angle to determine if rays are terminated according to[9]:

$$(\theta > \frac{\pi}{2} * \sigma) \tag{2.9}$$

Rays that are not terminated are emitted at the randomized angle. This means that rays with a greater inclination angle for θ are likelier to be terminated than lower values for θ.

2.6.2 Tracing an importance ray

An importance ray is launched at the eye and emitted through a pixel in the image plane. The ray has importance 1 from the beginning and this value will change through every intersection with a surface.

The probability $0 \leq \alpha \leq 1$ that a ray is terminated is compensated by the one-sample estimator in equation 2.10, where $F(\theta_{out}) = \pi\sigma cos\theta_{out} sin\theta_{out}$ and $\alpha = 1 - \sigma$. [7]

$$W(x \rightarrow \omega_{out}) = \frac{F(\theta_{out})}{1 - \alpha} W(x \leftarrow \omega_{in}) \tag{2.10}$$

The intersection points of the ray is stored in a tree. In each node of the tree the incoming importance, the reflected importance and the intersection surface is stored. The reflected ray and the importance that it carries is computed depending on the material of the surface as previously mentioned. The ratio between incoming importance and reflected importance correspond to the BRDF value and can be used to compute radiance when traversing the tree the opposite way [5].

2.6.3 Computing radiance

The radiance at the first intersection point is computed by traversing the ray tree backwards. The change in radiance between the incoming radiance $L(x \rightarrow \omega_{in})$ and the outgoing radiance $L(x \leftarrow \omega_{out})$ from a lambertian surface is the same ratio as between incoming and outgoing importance. At lambertian surfaces the direct light contribution is also added to the outgoing radiance. The radiance does not change when intersecting a perfect mirror since all light is reflected. The final radiance value is the radiance that is emitted from the top intersection node in the tree (i.e the first intersection of the importance ray).

[9] Mark Eric Dieckmann TNCG15: Advanced Global Illumination andRendering, Lecture 11 (Photon mapping 'light'), 2021

2.7 Converting radiance to pixel values

The radiance obtained at a pixel in the image plane is set as the color of that particular pixel. The values of the pixels are normalized by dividing the intensities with the maximum intensity in the image. For renderings where the intensity varies a lot, the square root of intensity values are used to set pixel values and find the maximum intensity. The values are also multiplied with 255.99 to produce values in the range 0 to 255 since the RGB values of the image are represented by 8 bits.

2.7.1 Average color

The integral for reflected light in the rendering equation 2.1 is computed over the hemisphere for all possible incoming directions, but since this is impossible to compute exactly we turn to numerical computations. In the Monte Carlo method a one sample estimator can be used to approximate the integral. The result produced is generally quite noisy.

The error decreases by $\frac{1}{\sqrt{N}}$ where N is the number of samples. When rendering the scene, different amount of samples for each pixel can be used. A high amount of samples can reduce the noise in the results [6].

2.7.2 Rendering bitmap images

In order to visualize the results, the library *EasyBMP* was used, which allows for generating simple bitmap images[10].

[10]Paul Macklin, EasyBMP project, http://easybmp.sourceforge.net/, retrieved 2021-11-09

3 Results

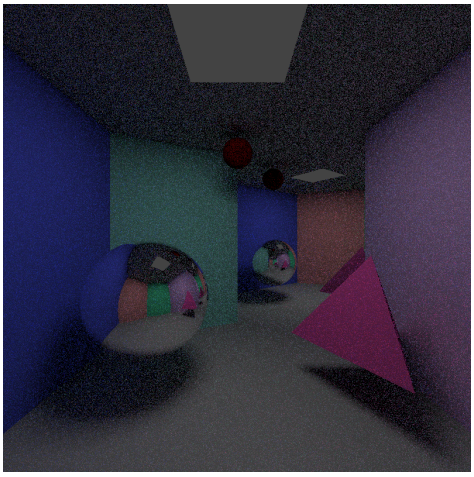
The implemented Monte Carlo ray tracer was used to render the scene described in section 2.1. The scene was rendered with 1 to 400 samples per pixel with a resolution of 800x800 pixels. The resulting rendered images are presented in Figure 3.1 and Figure 3.2. The rendering model has many parameters that can be changed but in the results below they are kept constant except for the number of samples.

The results show that the rendered image contains less noise the more samples are used. Figure 3.1 a) is very noisy and dark but Figure 3.2 f) contains barely any visible noise and have clear bright colors.

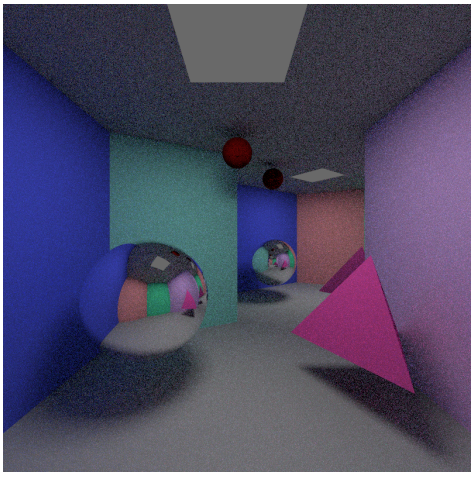
The scene rendered contains two mirrors, one mirror sphere and one mirror wall made from two triangle meshes. The mirrors seems to be accurate since they reflect the objects that are visible to it. Objects which are not visible in the original view plane directly can be observed indirectly in the perfect mirrors, see Figure 3.3. The mirrors does not change the amount of light that reflects onto its surface since it reflects all incoming light.

Color bleeding is visible at the purple wall to the right, see Figure 3.4, which receives pink light from the tetrahedron close to the wall and also in the roof above the red sphere which receives red light for the sphere. This is due to indirect light being transferred between surfaces according to 2.5.2.

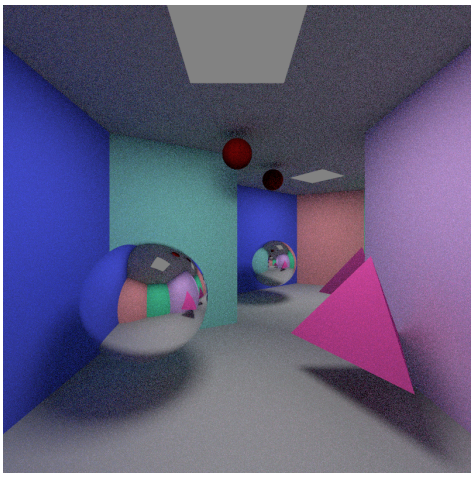
The shadows are soft due to the fact that an area light is used which means that edges of shadows will receive only part of the total radiance emitted from the light source. The areas in shadow still receive radiance from indirect illumination.



(a) 1 sample per pixel

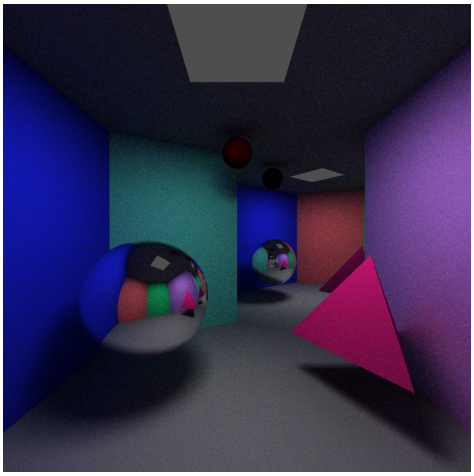


(b) 4 samples per pixel

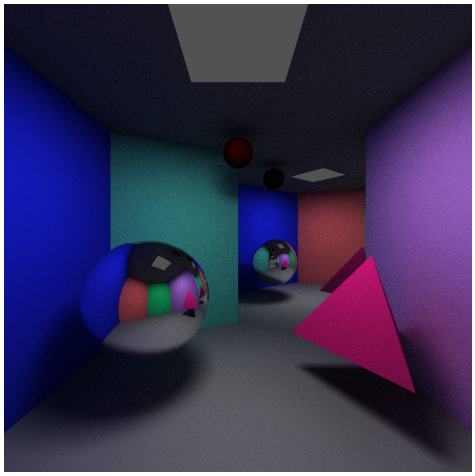


(c) 9 samples per pixel

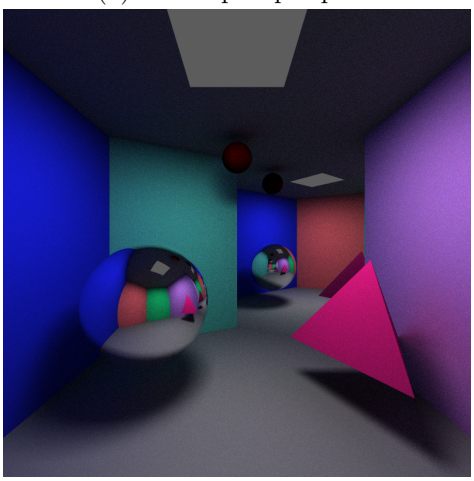
Figure 3.1: The scene rendered with 1, 4 and 9 samples per pixel respectively. The colors of the images are set to the square root of their original value since they would be very dark otherwise.



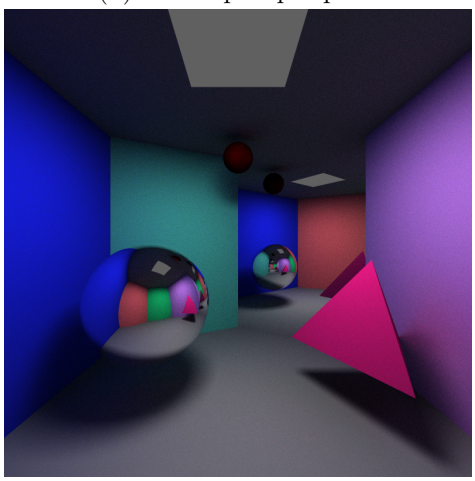
(a) 16 samples per pixel



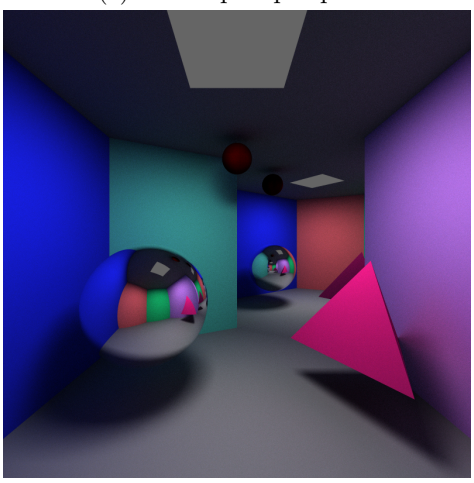
(b) 25 samples per pixel



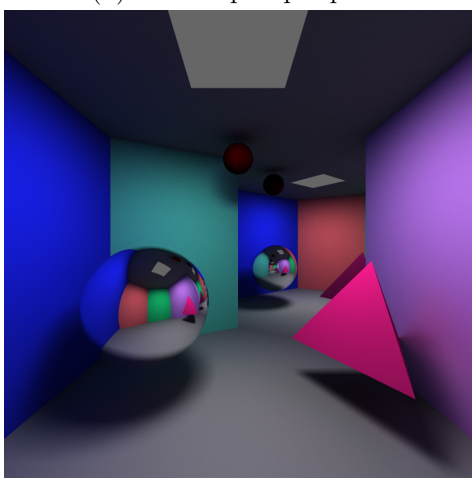
(c) 49 samples per pixel



(d) 100 samples per pixel



(e) 225 samples per pixel



(f) 400 samples per pixel

Figure 3.2: The scene rendered with 16, 25, 49, 100, 225 and 400 samples per pixel respectively.

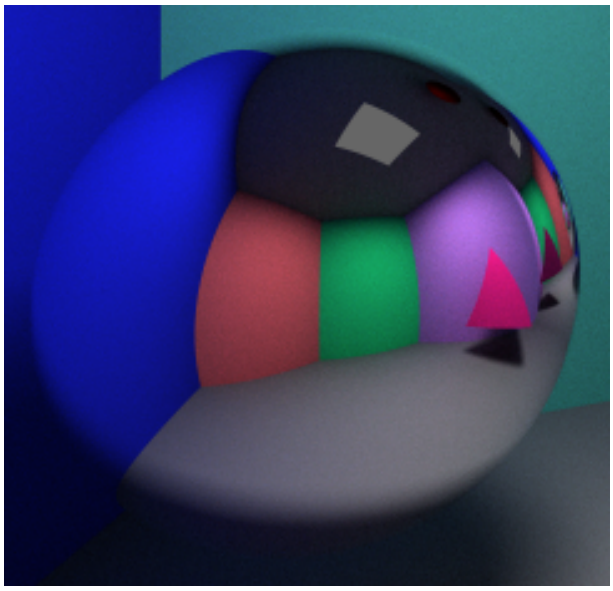


Figure 3.3: Close up on image rendered with 400 samples to give a better view on the objects reflected in the mirror sphere.

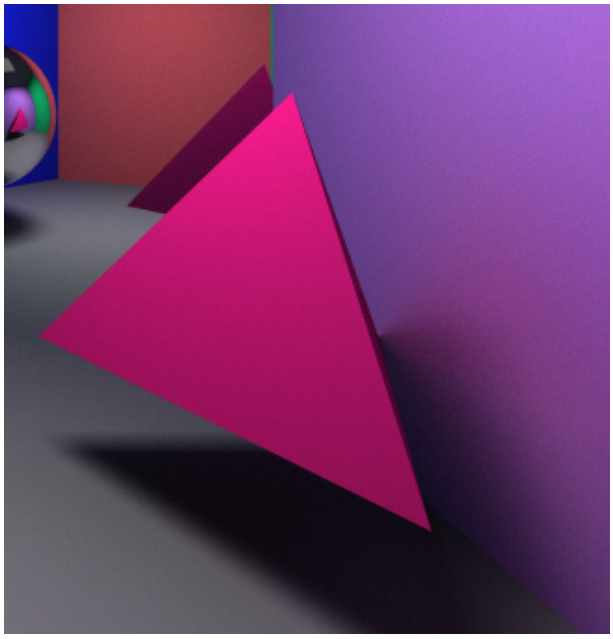


Figure 3.4: Close up on image rendered with 400 samples, which illustrates the color bleed that is present on the purple wall.

4 Discussion

This project resulted in a functioning Monte Carlo ray tracer which can handle both lambertian surfaces and perfect mirror surfaces. The scene receives light from an area light source which distributes light according to the distance between the surface and the light source. This creates brighter spots closer to the light. The roof is quite dark but that is a result from the computations of direct light at the points located on the roof triangles. When computing the direct light using the estimator in equation 2.7 the term $V(x, x_{L,i}) = 0$ which means that the shadow ray does not intersect with the triangles of the area light. This means that the roof points only receive indirect light.

If the scene would have been rendered without indirect light it would give a physically inaccurate representation of the world. This would not take into consideration that light bounces in the scene and that objects reflect light on each other. Using the approach of global illumination is therefore a much better representation of how light actually behaves and generates more realistically rendered images.

Having reasonable amounts of color bleeding is a good indication of a successful algorithm since too much or too little color bleed would indicate that the algorithm has errors regarding calculating indirect light.

4.0.1 Possible improvements

A possible issue with the implementation is the fact that ray paths are never terminated at perfect reflectors. Since the perfect reflectors do not absorb any importance, this means that theoretically rays could get stuck infinitely between mirror surfaces. This was however never something we encountered. Another issue would be if objects exist between the eye and camera plane since that would mean that the first intersected object lies behind the plane we wish to render.

Another possible improvement is regarding performance. Rendering times are quite long since the global illumination model runs on the *CPU* and no work has been done on multi-threading. Another possible improvement would be to research whether some parts of the model could be implemented to run on the *GPU* without completely re-implementing the whole code structure.

Ray randomization is an improvement that could be made. It would reduce the ordering

and aliasing effect and replace it by noise which is less disturbing[2].

Bibliography

[1] Tomas Akenine-Möller Henrik Wann Jensen. "The Race for Real-time Photorealism". In: *Sigma Xi Scientific Research Society, 2010.* (2010). URL: https://www.proquest.com/docview/504754186?.

[2] Jon Peddie. *Ray Tracing: A Tool for All.* Springer, Cham, 2019. ISBN: 978-3-030-17490-3.

[3] R. Keith Morley Peter Shirley. *Realistic Ray Tracing.* 2003. ISBN: 1568811985.

[4] Jason Rupard. "Ray Tracing and Global Illumination". In: (2003). URL: https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1100&context=ojii_volumes.

[5] Turner Whitted. "An improved illumination model for shaded display". In: (1980). URL: https://www.cs.drexel.edu/~david/Classes/Papers/Whitted80.pdf.

[6] Taiyo Wilson Stephen Merity Tegan Brennan. *Monte Carlo methods for improved rendering.* 2014. URL: https://smerity.com/montelight-cpp/.