Investigation of light pipe simulation algorithms

Anders Schou

Kongens Lyngby 2012
IMM-B.Sc-2012-01

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-B.Sc-2012-01

Summary (English)

The subject of this thesis is to test whether progressive photon mapping and/or stochastic progressive photon mapping can be used in daylight simulations of interior of a room with a light pipe introducing natural lighting into the room. The algorithms tested (progressive photon mapping and stochastic progressive photon mapping), are both multi-pass algorithms that solves global illumination in a robust way with bounded memory consumption, while converging to the correct solution.
To simulate the sky the CIE [DK02] standard sky patterns have been used which returns the luminance of an arbitrary sky element. In the tests a model of the VELUX TRM 010 Sun tunnel was used.

The results obtained from progressive photon mapping and stochastic progressive photon mapping have been tested against the analytical solution HOLIGILM [KDK08], and an implementation of path tracing.

Summary (Danish)

Emnet for denne afhandling er at teste om progressive photon mapping og/eller stochastic progressive photon mapping kan anvendes til daglys simuleringer af interiøret af et rum, hvor en lys-tunnel fører daglys ind. De testede algoritmer (progressive photon mapping og stochastic progressive photon mapping) er begge multi-pass algoritmer, der løser global belysning på en robust måde med begrænset hukommelsesforbrug, mens de konvergerer til den korrekte løsning.
Til at simulere himmeler er CIE [DK02] standard sky modeller blevet anvendt. Disse returnerer et vilkårligt himmelelements luminans. I testene er en model af VELUX TRM 010 sol-tunnel blevet anvendt.

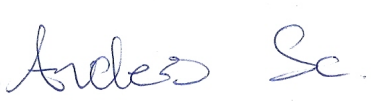
Resultaterne fra progressive photon mapping og stochastic progressive photon mapping er blevet testet imod den analytiske metode HOLIGILM [KDK08], og en implementering af path tracing.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring the B.Sc. degree in engineering. The project was proposed by VELUX A/S, and the project was developed in collaboration with VELUX A/S. This 15 ECTS credit points thesis was written under the supervision of Associate Professor Jeppe Revall Frisvad @ IMM DTU and Nicolas Roy from VELUX A/S.

The implementation in this project is based on the framework used in the course 02576 Physically based rendering.

Lyngby, 09-January-2012



Anders Schou

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

I would especially like to thank my supervisor Associate Professor Jeppe Revall Frisvad, without him this thesis would not be possible. I would also like to thank my secondary supervisor Nicolas Roy from VELUX A/S for being available, and offer information on the Sun Tunnel which was essential to the implementation.

I would like to thank Associate Professor Anders Lindbjerg Dahl for numerous suggestions and corrections throughout the project. Last but not least, I would like to thank Miroslav Kocifaj from Department of Interplanetary Matter Astronomical Institute of the Slovak Academy of Sciences for making the program HOLIGILM available to me.

Contents

Summary (English) i

Summary (Danish) iii

Preface v

Acknowledgements vii

1 Introduction 1

2 Previous work 3
2.1 A design tool for predicting the performance of light pipes 3
2.2 HOLIGILM: Hollow light guide interior illumination method-An analytic calculation approach for cylindrical light-tubes 4
2.3 Raytracing simulations for predicting light-pipe transmission . . 4

3 Sky model 7
3.1 Sampling of the sky model 9

4 The model 11
4.1 Collector and the pipe 11
4.2 Diffuser 12

5 Theory 13
5.1 The rendering equation 13
5.2 Path tracing 14
5.2.1 Bidirectional path tracing 14
5.3 Photon mapping 14
5.4 Flux emitted from a source 15

5.5 Progressive Photon Mapping 16
5.5.1 The algorithm 17
5.5.2 Implementation 18
5.6 Stochastic Progressive Photon Mapping 19
5.6.1 The algorithm 19
5.6.2 Implementation 20

6 Results **23**
6.1 Test PPM & SPPM vs Path Tracing 24
6.2 Test vs HOLIGILM 24
6.3 Lightpipe and CornellBox 28

7 Conclusion **33**

A PPM source code **35**
A.1 radiance estimate 35

B SPPM Source code **37**
B.1 Radiance estimate 37

C PPM original results **41**

D SPPM original results **49**

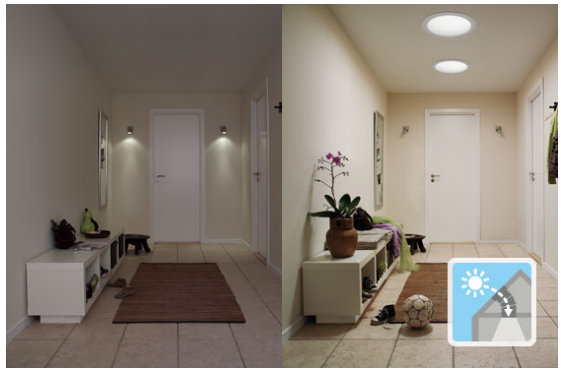
Bibliography **57**

Chapter 1

Introduction

This project has come from a suggestion from VELUX A/S. They where interested in simulation of their light pipe (the VELUX A/S product is called a sun tunnel). The application of a light pipe is to transport natural light from the outside, to the interior of a building/construction. This can be useful in small interior rooms, like a hall (Figure 1.1a and 1.1b) or a bathroom. It can also be used to reduce the artificial lighting required doing the day in offices, factories, computer labs etc.. Artificial lighting represents up to 30 % of the electrical energy consumption in commercial and office buildings [ORS00]. The use of natural lighting can reduce the energy consumption by up to 20-30 % [DS07]. The reduced need for artificial lighting is in itself a very important properties, especially with the focus on global warming [AZARS06] and CO_2 reduction, but economical consideration can also influence the use of these technologies.
A light pipe consist of a collector, a reflective tube, and a diffuser. The collector is collecting sunlight to the reflecting tube, this light then goes into the reflecting tube and then into the interior of the building via the diffuser that spreads the light.

Simulation and calculation on light pipes is not a new topic. There have been prototype- and full scale-experiments in different weather conditions and in different locations [DKK10, AZARS06, ORS00, PCC07], analytical solutions [KDK08,SS95] and simulations [DS07]. The denominator in the analytical methods and the simulations is, that they give the results in illuminance. What



(a) Before [weba] **(b)** After [webb]

Figure 1.1: Hall without and with light pipes installed

VELUX A/S is interested in, is a method to make a visualization of the room, as well as an illuminance and luminance estimate. The current Daylight visualizer doesn't handle specular reflections, which is needed for simulating a light pipe.

The approach used in this project is different from other simulations, as I intend to use progressive photon mapping and stochastic progressive photon mapping, as these methods are known the be robust methods. Additionally I would like to explore if these approaches could be used as a foundation the an implementation in VELUX's own program VELUX Daylight Visualizer.

Chapter 2

Previous work

2.1 A design tool for predicting the performance of light pipes

Jenkins, D., Muneer, T. and Kubie, J. [JMK05] have made a semi analytically and experimental model that predicts the performance of light pipes. This model assumes cylindrical light pipe and hemispherical collector. This method can give the illuminance on an arbitrary working plane, given the transmittance of the diffuser and collector and reflectance of the pipe. The diffuser in this model is non-uniform, and thus the model isn't accurate under clear sky conditions with a short light pipe. In this method the illuminance of an arbitrary point under the diffuser is given with

$$E_i = 0.494 \, \frac{\phi_{SP} \, cos^4(\theta)}{V^2}, \tag{2.1}$$

where ϕ_{SP} is the luminous flux of a straight pipe, θ is the angle between the line from diffuser to point of measurement and the normal of the diffuser and V is the vertical distance of pipe diffuser to point of measurement.
ϕ_{SP} is given with

$$\phi_{SP} = \tau_{\text{dome}} \, \tau_{\text{diffuser}} \, T_{\text{pipe}} \, E_{ex} \, \pi \, r^2, \tag{2.2}$$

where τ_{dome} τ_{diffuser} are the transmissions of the dome and diffuser, respectively, T_{pipe} is the transmission of a piece of pipe of unit aspect ratio, r is the radius of the pipe and E_{ex} is the external illuminance. This method clearly gives an even illuminace which contradicts the non-uniform diffuser. This is the reason this method isn't accurate under clear sky conditions with a short light pipe[1].

2.2 HOLIGILM: Hollow light guide interior illumination method-An analytic calculation approach for cylindrical light-tubes

HOLIGILM [KDK08], an analytical solution for straight and bended light pipes. HOLIGILM returns the illuminance at the workingplane/floor and just under the diffuser. Additionally HOLIGILM uses the 15 standard CIE sky-models. In HOLIGILM the collector is hemispherical, and the light-pipe is cylindrical. The diffuser is either perfect lambertian, transmitting or a mix of the two. The mix is an inner lambertian part, with an outer transmitting, or inner transmitting with an outer lambertian. This cover most light pipe systems, but not all. HOLIGILM can work with several light pipes of different properties in one simulation.

2.3 Raytracing simulations for predicting light-pipe transmission

Dutton and Shao [DS07] investigated and proved the accuracy of using raytracing to predict light pipe transmission. They used the program Photopia [2] to perform the simulations. According to the article, Photopia is limited to the tree I.E.S. sky models, namely overcast, partially cloudy and clear sky.
Their method take into account the complexity of the collector, but mentions nothing about the diffuser. To simulate the sky, they create a dome in the center of view. This dome is a patchwork of polygons, where each polygon is a lamp emitting parallel light into the center of the dome, additionally they have a small lamp emitting in one direction. The dome is simulations the diffuse light from the sky, while the lamp is simulations the direct light from the sun. The amount of light emitted from each polygon depend on the part of the sky model

[1] The longer the light-pipe, the closer the incoming light on the diffuser/transmitter will become diffuse.

[2] `http://www.ltioptics.com/Photopia/overview.html`

it represents.
As they use raytracing, the method isn't limited to any fixed model like cylindrical light pipes. As they are using raytracing, the method could be extended to create a visualisation of the room under the light pipe.

Chapter 3

Sky model

One of the important aspects of the project, is to have a good and physically correct sky-model. For this a standardised model has been used. This is the CIE sky model. It is described by five parameters a, b, c, d, and e. The model gives the relative luminance of an arbitrary sky element to the zenith luminance. In its very general form, the model looks like this [DK02]:

$$\frac{L_{\gamma\alpha}}{L_z} = \frac{f(\chi)\varphi(Z)}{f(Z_s)\varphi(0^\circ)} \tag{3.1}$$

Where $L_{\gamma\alpha}$ is the luminance of the arbitrary sky element γ and L_z is the zenith luminance.
The two functions, $f(\chi)$ and $\varphi(Z)$ is the scattering indicatrix function (3.2) and the luminance gradation function (3.3) respectively.

The scatting indicatrix function, relates the relative luminance of a sky element to its angular distance (χ) from the sun is given by [DK02]:

$$f(\chi) = 1 + c\left(\exp(d\,\chi) - \exp(d\,\pi/2)\right) + e\,\cos^2(\chi) \tag{3.2}$$

The luminance gradation function relates the luminance of a sky element to its

zenith angle.

$$\varphi(Z) = 1 + a \exp(b/\cos(Z)). \tag{3.3}$$

χ is given by

$$\chi = \arccos[\cos(Z_s)\cos(Z) + \sin(Z_s)\sin(Z)\cos(A_z)]. \tag{3.4}$$

In these equations (3.3)(3.2)(3.4) γ_s is the suns altitude, Z_s is the suns zenith angle given by $\frac{\pi}{2} - \gamma_s$, γ is the altitude of the sky element, Z is the zenith angle, α_s is the azimuth angle of the sun from north, α is the azimuth angle of the sky element, and $A_z = |\alpha - \alpha_s|$ Figure **??**.

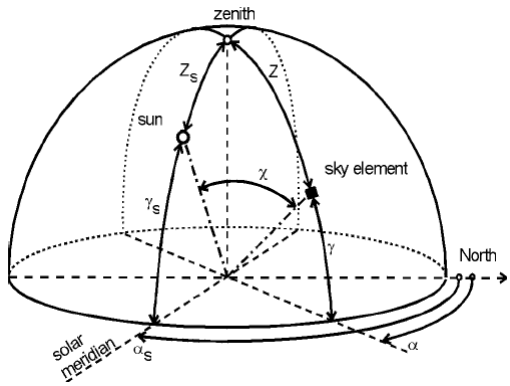


Figure 3.1: Angles defining the position of the sun and a sky element [DK02].

The parameters a, b, c, d and e describe the atmospheric conditions. There are 15 different standard sky parameters in the CIE, in [KDK08] type 12 is used. This is the CIE standard clear sky, with low illuminance turbidity. The parameters for this and other atmospheric conditions can be found in table 1 in [DK02] With no measurement of the zenith luminance L_z available, this can be calculated using (3.5).

$$L_z = A\sin(\gamma_s) + 0.7\,(T_v + 1)\,\frac{\sin(\gamma_s)^C}{\cos(\gamma_s)^D} + 0.04\,T_v. \tag{3.5}$$

This introduces some new parameters. T_v is the diffuse/sky horizontal illuminance normalized by its extraterrestrial value E_v. The parameter A, C, and D are characterising a certain sky type. If no measurement of T_v is available, a

typical value for T_v, and values for A, C, and D can be found in [DK02]. It should be noted that eq. 3.5 is not valid for sky model 1 to 6 where $T_v > 12$. For the sky model, the latitude and longtitude coordinates $55,46\,^\circ$ E and $12,30\,^\circ$ N have been used. This is approximately Kgs. Lyngby, found from [webc] which is the U.S. National Geospatial-intelligence Agency. These have been used to find the suns altitude and azimuth on September 15, 2011 at 3 PM from [webd] which is the Astronomical Applications Dept. U.S. Naval Observatory.

3.1 Sampling of the sky model

In most cases, the luminance from the sky model is far from uniform. To reduce the variance, and therefore the computation time, importance sampling has been used. The sampling method used is fairly simple. First the sky-model has been discretized into n points, then a 1D cdf for sampling each position have been calculated. The cdf for position $\mathbf{x}_i$ is given by:

$$\mathrm{cdf}(\mathbf{x}_i) = \frac{\sum_{j=1}^{i}(L(\mathbf{x}_j))}{\sum_{k=1}^{n}(L(\mathbf{x}_k))}, \tag{3.6}$$

where $L(\mathbf{x}_j)$ is the luminance for point j, so $L(\mathbf{x}_j) = L_{\gamma\alpha}$.
This cdf can then be used to sample a point on the hemisphere surrounding the model. The pdf for position $\mathbf{x}_i$ if just the cdf is used for sampling is then

$$\mathrm{pdf}(\mathbf{x}_i) = \frac{L(\mathbf{x}_i)}{\sum_{k=1}^{n}(L(\mathbf{x}_k))}. \tag{3.7}$$

Each of these n points actually represent a patch on the hemisphere. Right now $\mathrm{pdf}(\mathbf{x}_i)$ is the probability of sampling such a patch. The area of such a patch is given by

$$\mathrm{A}(\mathrm{P}_i) = \frac{2\pi r^2}{n}, \tag{3.8}$$

where r is the area of the sphere and P_i is patch i.
What is left, is to sample a position in this patch. Assuming uniform illumination over the patch in all directions, the illumination for each point on this patch is $L(\mathbf{x}_i)$. This enables us to sample a uniform position on the patch. The pdf for such a position is given by

$$\mathrm{pdf}(\mathrm{P}_i) = \frac{1}{\mathrm{A}(\mathrm{P}_i)} = \frac{n}{2\pi r^2}. \tag{3.9}$$

The actual pdf for a position on this patch is then

$$\mathrm{pdf}(\mathbf{x}_i) = \frac{L(\mathbf{x}_i)}{\sum_{k=1}^{n}(L(\mathbf{x}_k))}\frac{n}{2\pi r^2} \tag{3.10}$$

With the patch P_i chosen, the actual position $\mathbf{x}_i$ can then be sampled uniformly on a hemisphere, where the two random numbers ξ_1 and ξ_2 have been constrained accordingly.

Chapter 4

The model

The model used is a 3D model of the light pipe. The model Figure 4.1 is the standard TMR 010 - 10 inch rigid sun tunnel, and it is created and made public available from Google Schetchup by the Blue Marble Project and has been approved by VELUX [webe]. The model consist of 30902 triangles. For the model to work in the framework, it had to be converted from the Google Schetchup format to the wavefront format obj.

4.1 Collector and the pipe

The collector (Figure 4.2a) and the pipe are handled the same way. Given their probability of reflecting and refracting light a Russian roulette decide if the incoming light should be reflected, refracted or absorbed[1]. As the refraction index is unknown, the change of direction the refraction normally would cause have been disregarded. For the light pipe the probability for refraction is of course zero. With reflection, the outgoing direction is calculated as a perfect specular reflection.

[1] The absorption is given by $1 - \tau_t - \tau_r$, where τ_t and τ_r is the probability refraction and reflection respectively.

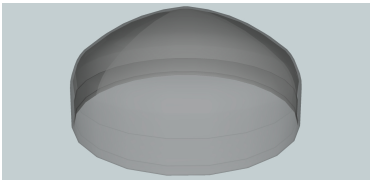
4.2 Diffuser

The diffuser Figure 4.2b is handled almost the same way as the collector and the pipe. A Russian roulette decides if the light that hits the collector should be transmitted or not. If the light is transmitted the a new direction is sampled in a cosine hemisphere around the direction $(x, y, z) = (0.0, -1.0, 0.0)$ as the normals on the on the models is not reliable. When the collector is used in the framework the BTDF[2] is given by

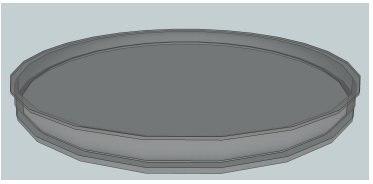
$$\text{BTDF} = \frac{\rho_t}{\pi} \cos(\theta), \tag{4.1}$$

where ρ_t is the transmittance coefficient, and θ is the angle between the outgoing and $(0.0, -1.0, 0.0)$.

Figure 4.1: Model of the TMR 010 light pipe/sun tunnel



(a) Model of the TMR 010 collector



(b) Model of the TMR 010 diffuser

Figure 4.2: TMR 010 collector and diffuser

[2]Bidirectional Transmittance Distribution Function

Chapter 5

Theory

5.1 The rendering equation

The rendering equation, introduced by Kajiya in 1986 [Kaj86] describes that radiance outgoing at each particular position and direction L_o is the sum of emitted radiance L_e and the reflected radiance L_r. In its integral from the rendering equation is:

$$L_0(\mathbf{x}, \vec{\omega}) = L_e(\mathbf{x}, \vec{\omega}) + \int_{2\pi} f_r(\mathbf{x}, \vec{\omega}', \vec{\omega})\, L_i(\mathbf{x}, \vec{\omega}')\, \cos(\theta') d\omega' \tag{5.1}$$

Where
$f_r(\mathbf{x}, \vec{\omega}', \vec{\omega})$ is the bidirectional reflectance distribution function BRDF, describing the reflected radiance from ω' to ω at position $\mathbf{x}$
$L_i(\mathbf{x}, \vec{\omega}')$ is radiance coming toward $\mathbf{x}$ at direction $\vec{\omega}'$
$\cos(\theta') = \vec{N_x} \cdot \vec{\omega'}$, $\vec{N_x}$ is the normal at position x

The brevity of the rendering equation contradicts the fact that it is in general impossible to solve analytically. The complexity comes from the physically based BSDF models, arbitrary scene geometry and intricate visibility relationships between objects [PH10].

5.2 Path tracing

Path tracing was the first unbiased Monte Carlo light transportation algorithm used in graphics. It was also introduced by Kajiya in the same paper that introduced the rendering equation [PH10]. The idea behind path tracing is when tracing rays, they don't terminate after a fixed amount reflections/refractions or at diffuse surfaces like in standard ray-tracing. In path tracing the rays are traced in a new random direction within the hemisphere of the object until a predefined depth, or it hits a light.

5.2.1 Bidirectional path tracing

Bidirectional path tracing is an accelerated version of the path tracing. Instead of just tracing rays from the eye, rays are also traced from the lights. Vaech and Guibas describe the method as "*These methods generate one subpath starting at a light source and another starting at the lens, then they consider all the paths obtained by joining every prefix of one subpath to every suffix of the other. This leads to a family of different importance sampling techniques for paths, which are then combined to minimize variance.*" [VG97]

5.3 Photon mapping

Photon mapping [Jen96] is a two pass global illumination algorithm. In the first pass photons from the light source and rays from the eye/camera are traced independently until some termination criterion is met. This can be number of bounces, or a non-specular surface has been hit. The photon are normally traced until they they are absorbed by the object they hit, or they hit a diffuse surface. In the second step they are connected to calculate a radiance value. The way they are connected is when a ray traced from the eye/camera hits a diffuse surface at $\mathbf{x}$ a lookup is made in the photon map. All the photon within a given range of $\mathbf{x}$ until upper bound of photons is then used in an radiance estimate to calculate the radiance at $\mathbf{x}$.

Unlike path tracing and bidirectional path tracing, photon mapping is a biased rendering algorithm. This bias is introduced in the radiance estimate as only a finite number of photons can be emitted and stored. The bias in photon mapping show itself as low frequency noise, which is less noticeable than the high frequency noise from path tracing and bidirectional path tracing. This

generally makes the results from photon mapping looks better if sufficient photons are used.

Given in mathematical terms, the exitant radiance at $\mathbf{x}$ can be estimated as

$$L(\mathbf{x}, \vec{\omega}) = \sum_{p \in \Delta A} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p) \frac{\Delta\phi_p(\mathbf{x}_p, \vec{\omega}_p)}{\Delta A}, \tag{5.2}$$

where f_r is the BRDF, $\vec{\omega}$ and $\vec{\omega}_p$ are the outgoing and incoming directions. The Δ-term is the irradiance estimate. This is given by

$$\Delta\phi_p(\mathbf{x}_p, \vec{\omega}_p) = \phi_p \, K\Big(\frac{\|\mathbf{x} - \mathbf{x}_p\|}{r}\Big), \tag{5.3}$$

where K is a filter kernel ϕ_p is the flux of the pth photon, r is the radius of the sphere containing the photons. The surface is assumed locally flat, so $\Delta A = \pi\, r^2$ as the photons are located on a disc.
With the simplest possible kernel and using n photons in the estimate, the exitant radiance can be expressed as

$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{n} \frac{f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p)\, \phi_p(\mathbf{x}_p, \vec{\omega}_p)}{\pi\, r^2} \tag{5.4}$$

The bias come when interpolating the n photons to estimate the exitant radiance at $\mathbf{x}$. Photon mapping is a consistent method, so as $n \to \infty$ and $r \to 0$ the results converges to the correct solution.

A method used to improve the result from photon mapping is called final gathering. Final gathering is a technique for estimating global illumination for a given point by sampling a number of directions in the hemisphere (or cosine weighted hemisphere) over that point. At these new points, an radiance estimate is used to calculate the global illumination.

5.4 Flux emitted from a source

Photons traced in photon mapping, progressive photon mapping and in stochastic progressive photon mapping are carrying flux. The flux Φ emitted from a source is given by

$$\Phi = \int_{2\pi}\int_{A} L_e \cos(\theta)\, \mathrm{d}A\, \mathrm{d}\omega = \int_{2\pi}\int_{A} L_e(\mathbf{x}, \vec{\omega})(\mathbf{n} \cdot \vec{\omega})\, \mathrm{d}A\, \mathrm{d}\omega \tag{5.5}$$

assuming $\mathbf{x}$, $\vec{\omega}$ are normalized.
The Monte Carlo estimate of the integral is

$$\Phi = \frac{1}{N} \sum_{i=1}^{N} \frac{L_e(\mathbf{x}_i, \vec{\omega_i})(\mathbf{n} \cdot \vec{\omega_i})}{\mathrm{pdf}(\vec{\omega}_i)\,\mathrm{pdf}(\mathbf{x}_i)}, \tag{5.6}$$

where $\mathrm{pdf}(\vec{\omega}_i)$ and $\mathrm{pdf}(\mathbf{x}_i)$ is the probability to sample direction $\vec{\omega}_i$ respectively position $\mathbf{x}_i$.
A photon is then a sample of this integral, so the flux of photon $\mathbf{p}$ is:

$$\Phi_p(\mathbf{y}, -\vec{\omega}) = \frac{1}{N} \frac{L_e(\mathbf{x}_i, \vec{\omega_i})(\mathbf{n} \cdot \vec{\omega_i})}{\mathrm{pdf}(\vec{\omega}_i)\,\mathrm{pdf}(\mathbf{x}_i)}, \tag{5.7}$$

and $\mathrm{pdf}(\vec{\omega}_i)$ is from sampling the bounding box/circle is given in [PH10]

$$\mathrm{pdf}(\vec{\omega}_i) = \frac{1}{2\pi(1 - \cos(\theta_{max})} \tag{5.8}$$

and $\mathrm{pdf}(\mathbf{x}_i)$ is is given by 3.10. With this the flux of photon $\mathbf{p}$ is:

$$\Phi_p(\mathbf{y}, -\vec{\omega}) = \frac{1}{N} \frac{4\pi\, L_e(\mathbf{x}_i, \vec{\omega_i})\,(\mathbf{n} \cdot \vec{\omega_i})}{L(\mathbf{x}_i)\, n} (1 - \cos(\theta_{max}) \sum_{k=1}^{n} (L(\mathbf{x}_k))\, r^2. \tag{5.9}$$

The factor $\frac{1}{N}$, is the normalization factor, and N is the number of photons emitted from the light source.

5.5 Progressive Photon Mapping

When solving the rendering equation, unbiased Monte Carlo ray tracing techniques is one of the most used. A problem with these methods is that they are not robust under all illumination conditions. For instance the caustics in a swimming pools. For these paths (specular-diffuse-specular) photon mapping is robust. The problem with photon mapping is that it is a biased method, and the trade-off from unbiased method is low frequency noise. To get an error free result from photon mapping an infinite number of photons have to be traced, and stored. This is not feasible or possible. Progressive photon mapping (PPM) solves this problem by progressive refinement [HJ09].
PPM is an algorithm developed by Hachisuka et al. [HOJ08] which keep the robustness of photon mapping, but converges asymptotically to the correct result with bounded memory.

5.5.1 The algorithm

Recalling 5.4 the radiance estimate

$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{n} \frac{f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p)\, \phi_p(\mathbf{x}_p, \vec{\omega}_p)}{\pi\, r^2}$$

If N photons are used in the photon map, but only N^β in the radiance estimate $\beta \in]0,1[$, as N becomes infinitely the radiance estimate L becomes [Jen01]

$$L(\mathbf{x}, \vec{\omega}) = \lim_{N\to\infty} \sum_{p=1}^{\lfloor N^\beta \rfloor} \frac{f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p)\, \phi_p(\mathbf{x}_p, \vec{\omega}_p)}{\pi\, r^2}. \tag{5.10}$$

As N becomes infinitely N^β becomes infinitely, but it will be infinitely smaller than N, which ensures that r will converge to zero. In normal photon mapping this is only of theoretical interest, as all the photons are stored. In PPM the requirements to this equation is furfilled without storing all the photons.

In PPM the average radiance value $L(\mathbf{x}, \vec{\omega})$ is estimated only at a single point $\mathbf{x}$. This estimate is done in each photon pass, and at the i-th pass, the radiance value at the position $\mathbf{x}$ in direction $\vec{\omega}$ is

$$L(\mathbf{x}, \vec{\omega}) \approx \frac{\tau_i(\mathbf{x}, \vec{\omega})}{N_e(i)\, \pi\, R_i(\mathbf{x})^2}, \tag{5.11}$$

where $\tau_i(\mathbf{x}, \vec{\omega})$ is the accumulated unnormalized flux times the bidirectional reflectance distribution function (BRDF), $R_i(\mathbf{x})$ is the search radius and $N_e(i)$ is the number of emitted photons after i passes. $N_e(i)$ is normally proportional to i i.e., the number of photons emitted per pass is constant. The updating procedure for $\tau_i(\mathbf{x}, \vec{\omega})$ and $R_i(\mathbf{x})$ is given in (5.12) and (5.13).

$$R_{i+1}(\mathbf{x}) = R_i(\mathbf{x}) \sqrt{\frac{N_i(\mathbf{x}) + \alpha M_i(\mathbf{x})}{N_i(\mathbf{x}) + M_i(\mathbf{x})}} \tag{5.12}$$

$$\tau_{i+1}(\mathbf{x}, \vec{\omega}) = (\tau_i(\mathbf{x}, \vec{\omega}) + \Phi_i(\mathbf{x}, \vec{\omega}))\, \frac{R_{i+1}(\mathbf{x})^2}{R_i(\mathbf{x})^2}, \tag{5.13}$$

where $M_i(\mathbf{x})$ photons are found within the search radius during the i-th pass, and $N_i(\mathbf{x})$ is the accumulated photon count. The updating procedure for $N_i(\mathbf{x})$ is

$$N_{i+1}(\mathbf{x}) = N_i(\mathbf{x}) + \alpha M_i(\mathbf{x}), \tag{5.14}$$

and the the updating procedure for $\Phi_i(\mathbf{x}, \vec{\omega})$ is

$$\Phi_i(\mathbf{x}, \vec{\omega}) = \sum_{p=1}^{M_i(\mathbf{x})} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}_p)\, \Phi_p(\mathbf{x}_p, \vec{\omega}_p). \tag{5.15}$$

In this α is a user-defined parameter in the interval $(0, 1)$, f_r is the BRDF and $\Phi_p(\mathbf{x}_p, \vec{\omega}_p)$ is the unnormalized flux of photon p and $\vec{\omega}_p$ is the incoming direction of the photon [HOJ08].

5.5.2 Implementation

In the original paper of PPM [HOJ08] it was suggested to use the data-structure of

```
struct hitpoint {
 position x    Hit location
 normal n      Normal at x
 vector omega  Ray direction
 integer BRDF  brdf at x
 float x,y     Pixel location
 color wgt     Pixel weight
 float R       Current photon radius
 integet N     Accumulated photon count
 color tau     Accumulated reflected flux
}
```

In my implementation of PPM i use the following data-structure. It is very similar to the data-structure in [HOJ08], the only differences is that I keep the results of the radiance estimate, so a visualisation can be made after each photon tracing pass, and a few data types have been changed to match the framework. The implementation of the updating procedures can be seen in Appendix A

```
struct hitpoint {
 position x            Hit location
 direction omega       Ray direction
 normal n              Normal at x
 vector BRDF           BRDF at position x
 float x,y             Pixel footprint.
 float wgt             Weight at x
```

```
 float R                Search radius
 integer N              Accumulated photon count
 vector tau             Accumulated flux
 vector power           This can be used directly in the visualisation
}
```

5.6 Stochastic Progressive Photon Mapping

A problem with PPM is that it is not efficient at rendering distributed ray tracing effect, such as anti aliasing, motion blur etc.. This is effects where the radiance have to be calculated over a region. Stochastic progressive photon mapping solves this problem, while keeping the robustness of progressive photon mapping (and of course photon mapping) [HJ09].

In stochastic progressive photon mapping rays are traced from the eye and stored at the first non-specular surface. After this photons are emitted from the light source. If they hit within some region of the stored positions they are used to estimate to radiance; after the photon pass and each following photon pass, distributed rays are traced and the hit points are stored. These new hit positions are then used in the radiance estimate in the following photon pass. This way the estimate gets progressively better, see Figure 5.1.

5.6.1 The algorithm

Stochastic progressive photon mapping (SPPM) is a method to calculate to average radiance value $L(S, \vec{\omega})$ over a region S. SPPM is like PPM a multi-pass method to solve the rendering equation.

In SPPM the updating procedure is almost the same as in PPM. Instead of using a fixed position $\mathbf{x}$, SPPM used randomly generated positions $\mathbf{x}_i$ in the region S. The positions $\mathbf{x}_i$ is generated by distributed ray tracing. So the updating procedure looks like this:

$$L(S, \vec{\omega}) \approx \frac{\tau_i(S, \vec{\omega})}{N_e(i)\,\pi\,R_i(S)^2} \tag{5.16}$$

$$R_{i+1}(S) = R_i(S)\sqrt{\frac{N_i(S) + \alpha M_i(\mathbf{x}_i))}{N_i(S) + M_i(\mathbf{x}_i)}} \tag{5.17}$$

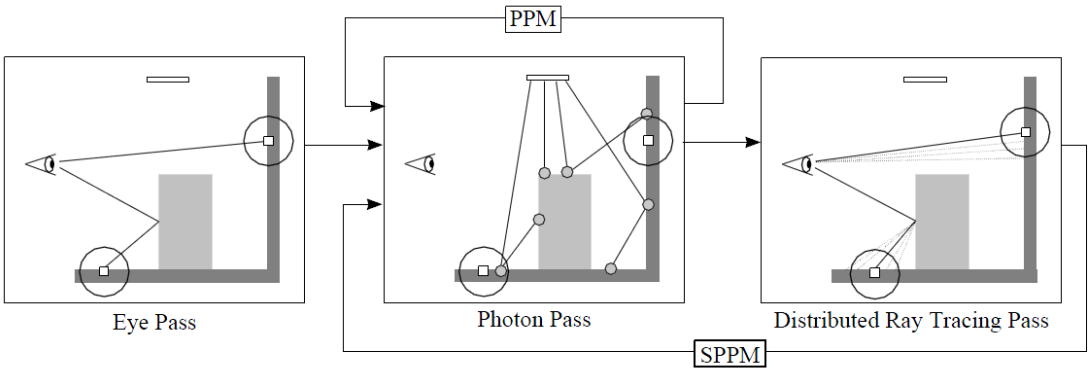


Figure 5.1: "*Difference between the algorithms of progressive photon mapping (PPM) and stochastic progressive photon mapping (SPPM). In order to compute the average radiance values, SPPM adds a new distributed ray tracing pass after each photon tracing pass. The photon tracing algorithm itself stays the same, but PPM uses a fixed set of hit points (shown as squares), whereas SPPM uses randomly generated hit points by the distributed ray tracing pass.*" [HJ09]

$$\tau_{i+1}(S,\vec{\omega}) = (\tau_i(S,\vec{\omega}) + \Phi_i(\mathbf{x}_i,\vec{\omega}))\,\frac{R_{i+1}(S)^2}{R_i(S)^2} \tag{5.18}$$

$$N_{i+1}(S) = N_i(S) + \alpha M_i(\mathbf{x}_i) \tag{5.19}$$

$$\Phi_i(\mathbf{x}_i,\vec{\omega}) = \sum_{p=1}^{M_i(\mathbf{x}_i)} f_r(\mathbf{x}_i,\vec{\omega},\vec{\omega}_p)\,\Phi_p(\mathbf{x}_p,\vec{\omega}_p). \tag{5.20}$$

5.6.2 Implementation

The data structure used for SPPM is almost identically to the one used in PPM, there are two modification. This is that the origin of the ray and the hit position from the distributed ray-tracing pass is troed. This is used in used in calculating $\cos(\theta_{max})$, that is used in the distributed ray-tracing pass.
So the data structure is,

```
struct hitpoint {
```

```
 position x            Hit location
 direction d           Ray direction
 position x2    Secondary hit location
 origin o              Origin (last specular hit)
 normal n              Normal at position
 float BRDF            BRDF at position
 float x,y             Pixel footprint.
 float wgt             Weight at position x
 float R               Search radius
 integer N             Accumulated photon count
 vector tau            Accumulated flux
 vector power          This can be used directly in the buffer
}
```

In the distributed ray-tracing pass, the directions are sampled in a cone around the original direction, with the same sampling procedure used when sampling a direction from the environment map to the model.

Chapter 6

Results

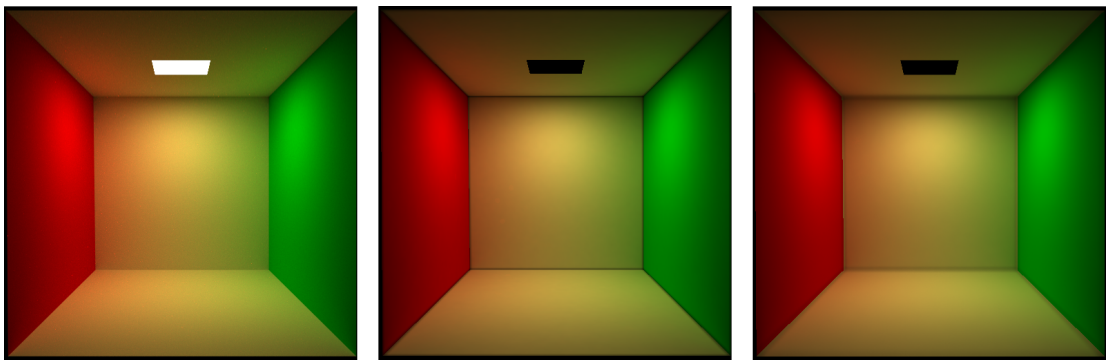


Figure 6.1: Results from Path tracing, progressive photon mapping and stochastic progressive photon mapping. The render time is approximately two hours. Path tracing have used 751 iterations, PPM have used 4242 iterations and SPPM have used 4151 iterations.

6.1 Test PPM & SPPM vs Path Tracing

To test the implementation of PPM and SPPM, I've compared the results of an implementation of path tracing with the results from PPM and SPPM. The scene used to test them on, is a standard Cornell Box. The way the results have been compared is that I have taken the euclidean distance between the tree radiance values in the results from path tracing and PPM, and the euclidean distance between path tracing and SPPM. The tree methods have been running approximately two hours. See Figure 6.1 for the results.

If we disregard the error introduced by the missing implementation/visualization of the light source, then the largest error is found in the boundary. This error is introduced because the surface assumed locally flat, and the radiance estimate only look for photons in a circle around the normal. So when dividing with the area in equation 5.11 the area is to large, as some of the circle will be off the edge of this surface, and therefore photons can't land on it.
The distance between path tracing and PPM is in the interval $[3 \cdot 10^{-6}, 2.3987]$ and the mean distance is 0.0638 after the error from the light source is removed, and 0.3866 before. Between path tracing and SPPM the distance is in the interval $[1.5 \cdot 10^{-5}, 2.2823]$ and the mean distance is 0.0629 after the error from the light source is removed, and 0.3859 before.
The difference between each of the tree colors have also been testes as the path traced image looks brighter than both PPM and SPPM. In all the colors, the largest deviation is still in the boundaries. When comparing the deviation in the colors, it is the red color that have the largest deviation from the path traced result, with a mean difference of 0.0501 in PPM and 0.0491 in SPPM.

The distance and difference can be seen in Figure 6.2 for PPM results and 6.3 for SPPM results.

6.2 Test vs HOLIGILM

Holigilm is the analytical tool used to test these methods against. The reason for taking Holigilm is the possibility to change the settings to match the set-up with the one i use as mush as possible. It is possible to set room, light-pipe and to some degree sky-model properties.

As mentioned in section 2.2 the results given by HOLIGILM is in illuminance.

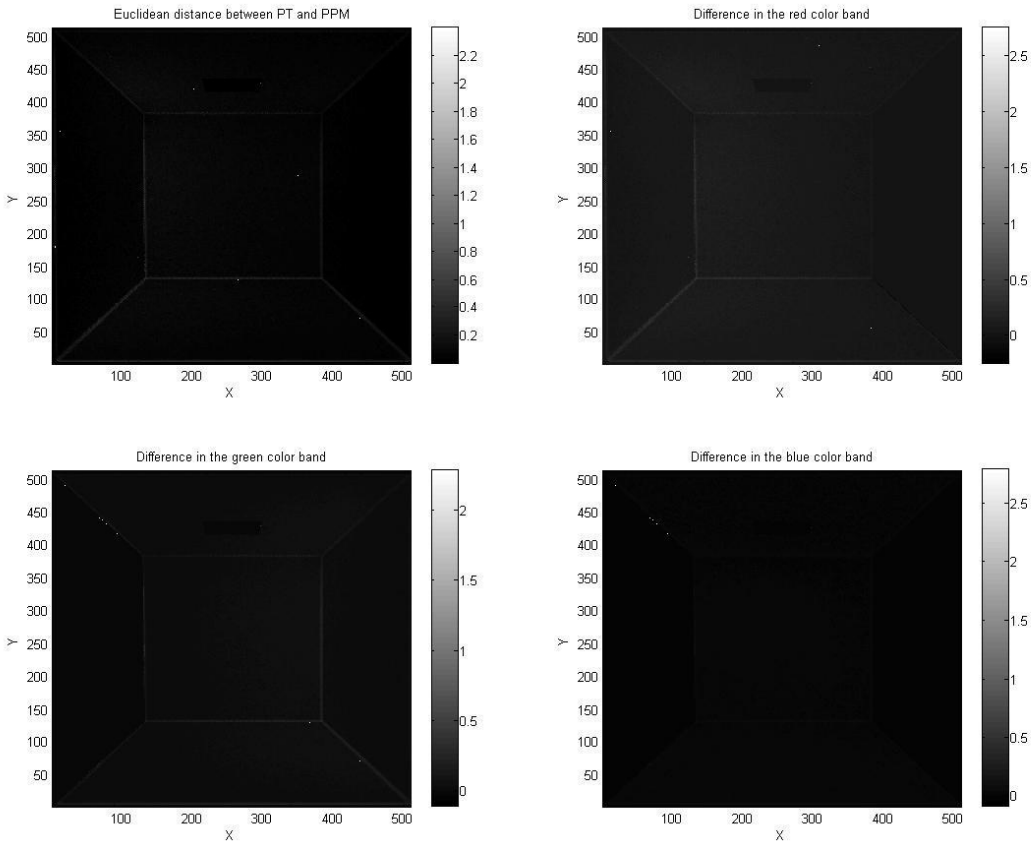


Figure 6.2: Showing the euclidean distance and the difference between the results from path tracing and progressive photon mapping. Top left: euclidean distance, top right: difference in red, bottom left: difference green, bottom right: difference blue. The bright dots are artefacts introduced by MatLab, when viewing the plots from this specific direction.

To compare the results from PPM and SPPM with those from HOLIGILM, the radiance estimate have to be changes to an illuminance estimate. Only two changes that need to the made to the framework, to trace radiance or luminance is the same, so the first change is to remove the conversion of luminance to radiance[1]. Second, the BRDF have been removed from the radiance estimate, this changes it to an irradiance estimate, and as we then trace luminance through the scene, it works as an illuminance estimate. The results from framework can

[1] Remember that the sky model original gives luminance.

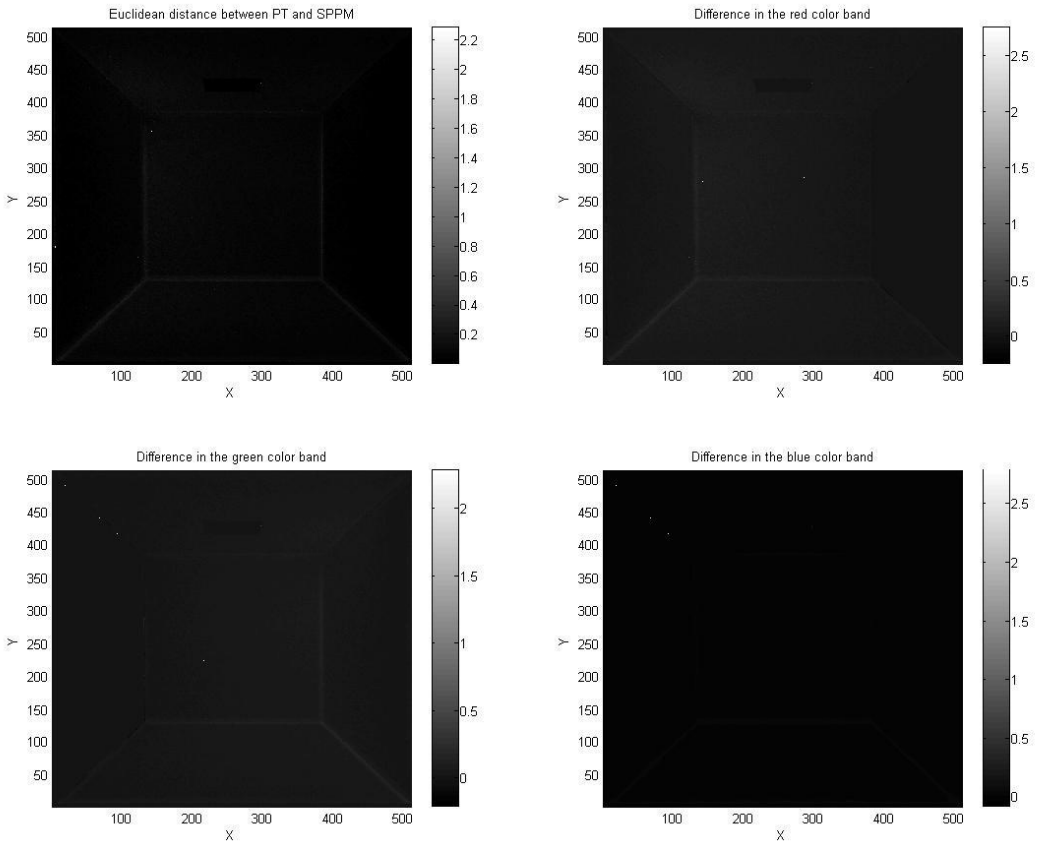


Figure 6.3: Showing the euclidean distance and the difference between the results from path tracing and stochastic progressive photon mapping. Top left: euclidean distance, top right: difference in red, bottom left: difference green, bottom right: difference blue. The bright dots are artefacts introduced by MatLab, when viewing the plots from this specific direction.

be seen in Figure 6.4 and Figure 6.5. When comparing these results with the output from Holigilm Figure 6.6 the main difference is the values. The values given by the illuminance estimate by PPM and SPPM are higher than the values given by Holigilm but the illuminance distribution is very similar to each other. This was to be expected as the diffuser is both the framework and Holigilm is lambertian. When looking at the values given by the framework the mean illuminance is 34.9053 lux in PPM and 35.4292 lux in SPPM. Both these values are higher than the 29.7978 lux given by Holigilm. In PPM the highest value

is 153.9830 lux located at $(x, y) = (254, 308)$. In SPPM it is 161.0490 lux and located at $(x, y) = (139, 254)$[2]. In Holigilm the highest illuminance value 40.2 lux and is located right under the light pipe, this is not the case in either PPM or SPPM. It is expected to "move" to the correct location as the number of iteration increases. The illuminance estimate used 200 iterations in PPM with 100000 photons per iteration and 209 iterations in SPPM, also with 100000 photons per iteration.

The variation between PPM and SPPM is expected. They have been running a different number of iterations and they converge at different rates to the correct solution, so as the variation is only 0.5239 lux which is approximately 1.5 % in the mean illuminance it is nothing that attracts attention as there haven't been preformed more iterations and variation in the results from the two methods is expected. The different between PPM and HOLIGILM and SPPM and HOLIGILM is more interesting. There could be several reasons for this difference. The difference in the models or difference in the zenith luminance used in equation 3.1. In the model the main difference is the collector, and especially the diffuser. In HOLIGILM, this is a disc, while in the model of Velux sun tunnel it is a 3D geometric object.

With the difference in the models and the exact number behind HOLIGILM is unknown, this is as good a result as one can hope for. The way the illuminance is distributed is the same in all methods, the only variation is the values, and as these are almost the same in PPM and SPPM, this is a satisfying result.

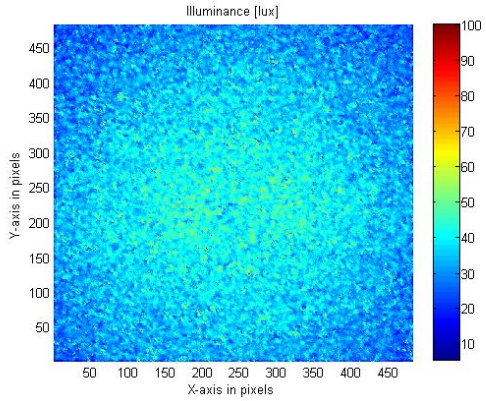


Figure 6.4: PPM illuminance estimate. The values have an upper bound of 100. Looking up at the floor.

[2]This is expected to even out as more iteration is used. But it could take a long time with the scene size vs. the resolution.

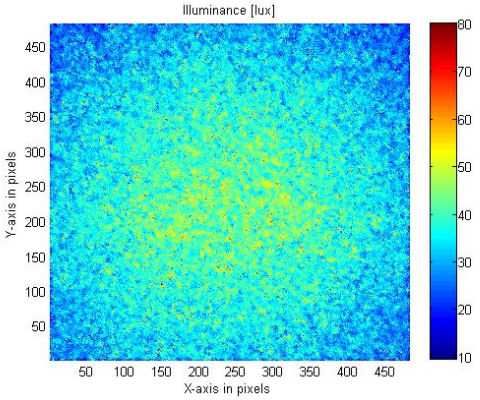


Figure 6.5: SPPM illuminance estimate. The values have an upper bound of 100. Looking up at the floor.

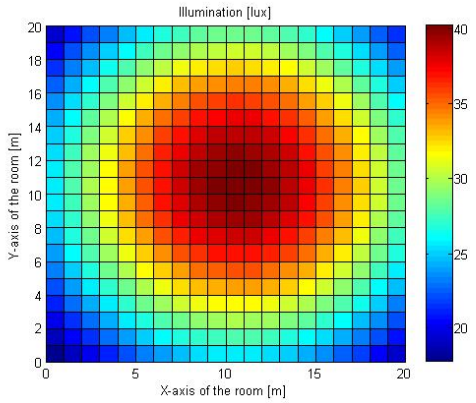


Figure 6.6: HOLIGILM results. The data have been transferred to MatLab to get the same colormap as Figure 6.4 and Figure 6.5. The setup in HOLIGILM matches the setup used in the framework as much as possible.

6.3 Lightpipe and CornellBox

The implementation of these methods have the possibility to create a visual representation of the room, as an radiance estimate is calculated. This is what other methods could be seem missing. Previous work have been about calculating the illuminance of the surfaces of the room. If one is only interested in the illuminace and the constrains of the other method isn't a problem, these

methods are probably to be preferred, as they are a lot faster than PPM and SPPM. On the other hand, PPM and SPPM are not limited to cylindrical light pipes with hemispherical collector and a lambertian diffuser[3]. Any model and any number of light pipe can be used. Additionally PPM and SPPM can also be used to give a visualization of the room.

In both the implementation of PPM and SPPM I used 100000 photon per iteration, a weight of $\alpha = 0.8$ and an initial radius of $r = 0.3$. The results after 1, 4, 16, 64, 256 and 1024 iterations can be seen in Figure 6.7 and Figure 6.8. The scene have the dimensions $80 \times 80 \times 80$ inch and the 10 inch lightpipe described in chapter 4 have been used. The figures 6.7 and 6.8 shows the progressive nature of the two methods. It looks like there could be a bug in the model with the bright ring in the celling around the diffuser, or it is energy from the initial radius. The later is unlikely as the ring doesn't fade out as one would have expected.

[3] If a non-lambertian transmitter is used, small changes will have to be made to the framework

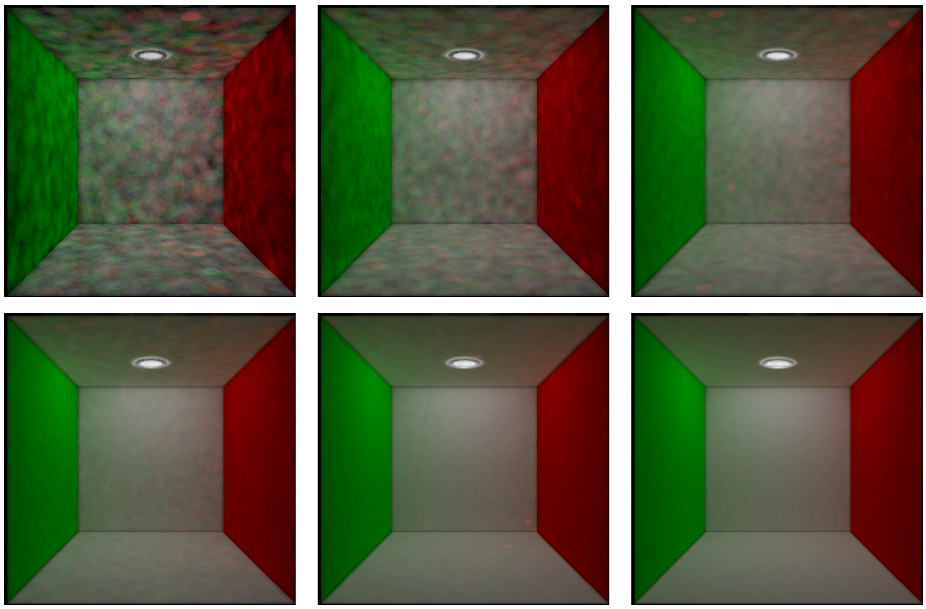


Figure 6.7: 80 x 80 inch CornellBox with 10 inch light pipe, rendered using PPM with respectively 1, 4, 16, 64, 256 and 1024 photon tracing passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8. The values in the original BMP-files have been doubled to make the images more clear. The original images can be seen in Appendix C in large scale.

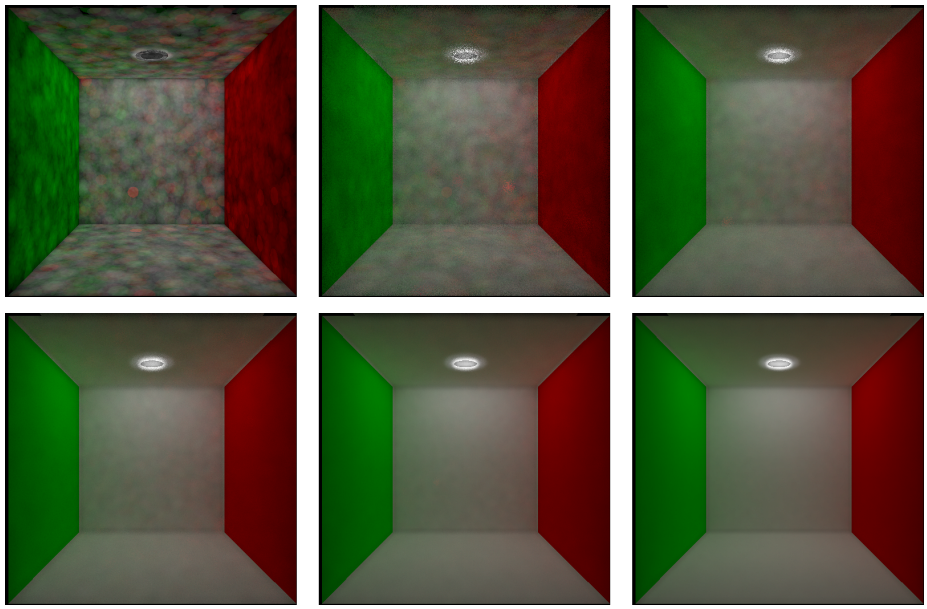


Figure 6.8: 80 x 80 inch CornellBox with 10 inch light pipe, rendered using SPPM with respectively 1, 4, 16, 64, 256 and 1024 photon tracing passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8. The values in the original BMP-files have been doubled to make the images more clear. The original images can be seen in Appendix D in large scale.

Chapter 7

Conclusion

The primary focus of the work presented in this theses, have been on the implementation of PPM and SPPM to test whether these can be used the estimate the illuminance and radiance of light pipe systems. Another aspect of the thesis have been in implement the CIE sky models and use this as light sources to emit photons from. To test the illuminance estimate the result have been compared to analytical solution provided by HOLIGILM which also uses the CIE sky-models. To test the radiance estimate, the implementations have been tested against an implementation of path tracing in a standard Cornell Box with an area light source.

The method used in the thesis shows that it is possible to use PPM and SPPM to get a radiance estimate, but also to get an illuminance estimate with a light pipe system. When the illuminance and BRDF and/or radiance is known, a luminance estimate is easy to obtain.

When the result from the illuminance estimate has been compared to the result from HOLIGILM, it shown that the illuminance from PPM and SPPM are generally higher than what HOLIGILM return. This is expected as the model used in the framework is different from the one used in HOLIGILM. Also the zenith luminance used in the sky model could be different.
The interesting part is that the illuminance distribution look similar, this, together with the low difference in the radiance values when PPM and SPPM is

compared with path tracing shows that PPM and SPPM can be used for estimating the radiance and illuminance of the interior of a room with a light pipe installed.

PPM and SPPM are both relativity slow methods when implemented only on the CPU, and would go under the so-called off-line methods, as the calculations isn't done in real time. Possible improvements could be:

- Use a simpler model. The one currently used consists of 30902 triangles.
- Use the extensive computational power available on the GPU.
- Use a faster (perhaps the analytical solutions) to estimate the how much light is emitted from the diffuser, and the use an area light source instead of the CIE models. (One would lose the generality SPPM and PPM gives regarding the model of the light pipe.)

All of the tree above suggestions will reduce the time an iteration takes. If it can lead to real time rendering, further work will show.
While the CPU implementation of PPM and SPPM are not suitable for something like Daylight Visualizer, it could be used to test new designs of light pipes, as both PPM and SPPM works with any model. The only thing needed is the BTDF for the diffuser, index of refraction for the collector and pipe or the percentage of light reflected and refracted.

Appendix A

PPM source code

Only the actual code needed for the updating procedure in PPM is shown here.

A.1 radiance estimate

```
1  void PhotonMap::radiance_estimate(int emitted, Photon &phot)
2  {
3
4    NearestPhotons np;
5    np.dist2 = (float*)alloca(sizeof(float)*(half_stored_photons+1));
6    np.index = (Photon**)alloca(sizeof(Photon*)*(half_stored_photons
         +1));
7
8    np.pos[0] = phot.pos[0];
9    np.pos[1] = phot.pos[1];
10   np.pos[2] = phot.pos[2];
11
12   np.max = half_stored_photons;
13   np.found = 0;
14   np.got_heap = 0;
15   // Set some value for the maximum distance to look for diffuse
         hit.
16   np.dist2[0] = phot.R*phot.R;
17
18   // locate the nearest photons
```

```
19    locate_photons( &np, 1 );
20
21    if(np.found>0)
22    {
23      int M = 0; CGLA::Vec3f temp_flux(0.0f);
24      for(int idx=1;idx<=np.found;idx++)
25      {
26        Photon *p = np.index[idx]; // The nearest photon.
27        CGLA::Vec3f dir = photon_dir(p);
28        if(CGLA::dot(dir,phot.normal)>0.0f &&
29        fabs(CGLA::dot(p->pos-phot.pos,phot.normal))<0.01)
30        {
31          M++;
32          temp_flux+=p->power;
33        } //end if
34
35      } // end for
36      float new_R = phot.R;
37
38      if(M>0)
39      {
40        // (N+a*M)/(N+M)
41        float scale = (phot.N+phot.w*M)/(phot.N+M);
42        new_R *= std::sqrt(scale);
43        temp_flux *= phot.brdf;
44        phot.flux = (phot.flux+temp_flux)*scale;
45
46      }
47      phot.power = 1/(static_cast<float>(M_PI)*phot.R*phot.R)*(phot.
            flux/static_cast<float>(emitted));
48      phot.N += phot.w*M;
49      phot.R = new_R;
50    } // end if
51  } // end radiace_estimate
```

Appendix B

SPPM Source code

Only the actual code needed for the updating procedure in SPPM is shown here.

B.1 Radiance estimate

```
1  void PhotonMap::radiance_estimate(int emitted, Photon &phot, bool
       first)
2  {
3
4    NearestPhotons np;
5    np.dist2 = (float*)alloca(sizeof(float)*(half_stored_photons+1));
6    np.index = (Photon**)alloca(sizeof(Photon*)*(half_stored_photons
         +1));
7
8    // Selection hit point.
9    if(first)
10   {
11     np.pos[0] = phot.pos[0];
12     np.pos[1] = phot.pos[1];
13     np.pos[2] = phot.pos[2];
14   }
15   else
16   {
17     np.pos[0] = phot.second_pos[0];
18     np.pos[1] = phot.second_pos[1];
```

```
19      np.pos[2] = phot.second_pos[2];
20    }
21
22    np.max = half_stored_photons;
23    np.found = 0;
24    np.got_heap = 0;
25    // Set some value for the maximum distance to look for diffuse
          hit.
26    np.dist2[0] = phot.R*phot.R;
27    // locate the nearest photons
28    locate_photons( &np, 1 );
29    CGLA::Vec3f pos;
30    if(first)
31    {
32       pos[0] = phot.pos[0];
33       pos[1] = phot.pos[1];
34       pos[2] = phot.pos[2];
35    }
36    else
37    {
38      pos[0] = phot.second_pos[0];
39      pos[1] = phot.second_pos[1];
40      pos[2] = phot.second_pos[2];
41    }
42
43    if(np.found>0)
44    {
45      int M = 0; CGLA::Vec3f temp_flux(0.0f);
46      for(int idx=1;idx<=np.found;idx++)
47      {
48        Photon *p = np.index[idx]; // The nearest photon.
49        CGLA::Vec3f dir = photon_dir(p);
50
51        if(CGLA::dot(dir,phot.normal)>0  &&
52        fabs(CGLA::dot(p->pos-pos,phot.normal))<0.01)
53        {
54          M++;
55          temp_flux+=p->power;
56        } //end if
57
58      } // end for
59      float new_R = phot.R;
60
61      if(M>0)
62      {
63        // (N+a*M)/(N+M)
64        float scale = (phot.N+phot.w*M)/(phot.N+M);
65        new_R = phot.R*std::sqrt(scale);
66        temp_flux *= phot.brdf;
67        phot.flux = (phot.flux+temp_flux)*scale;
68
69      }
70      phot.power = 1/(static_cast<float>(M_PI)*phot.R*phot.R)*(phot.
            flux/static_cast<float>(emitted));
71      phot.N += phot.w*M;
```

```
72          phot.R = new_R;
73      } // end if
74  }
```

Appendix C

PPM original results

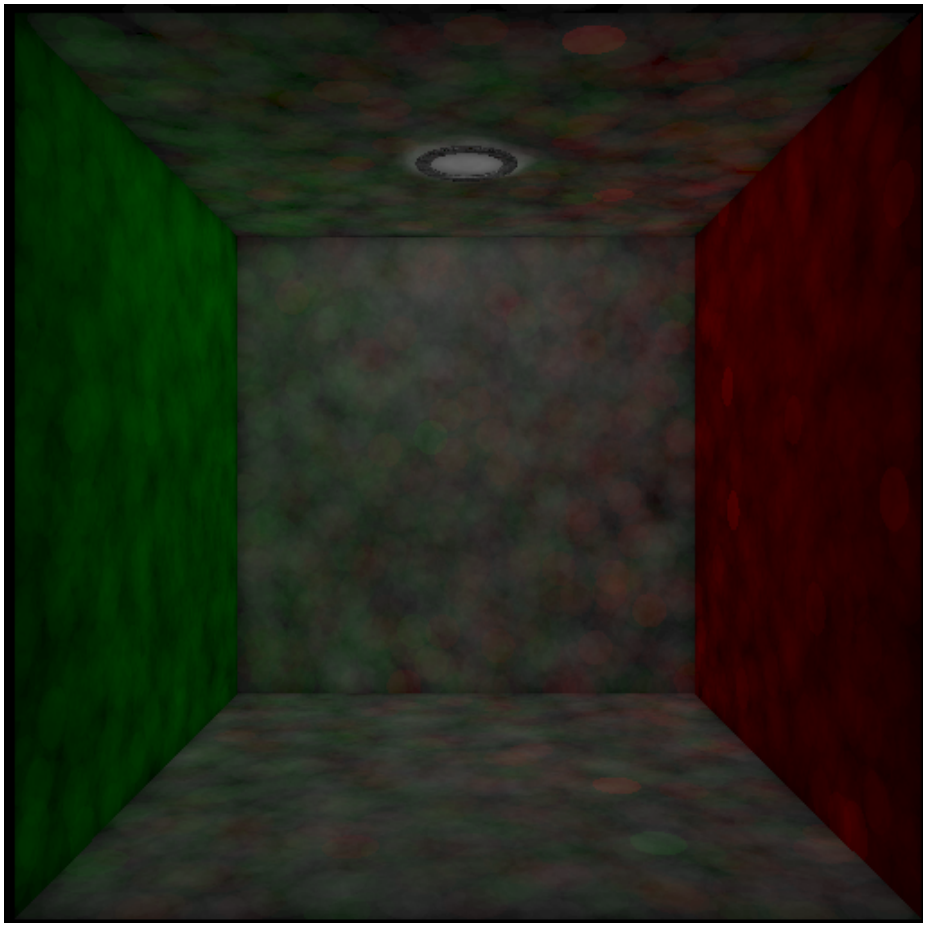


Figure C.1: PPM after one pass with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

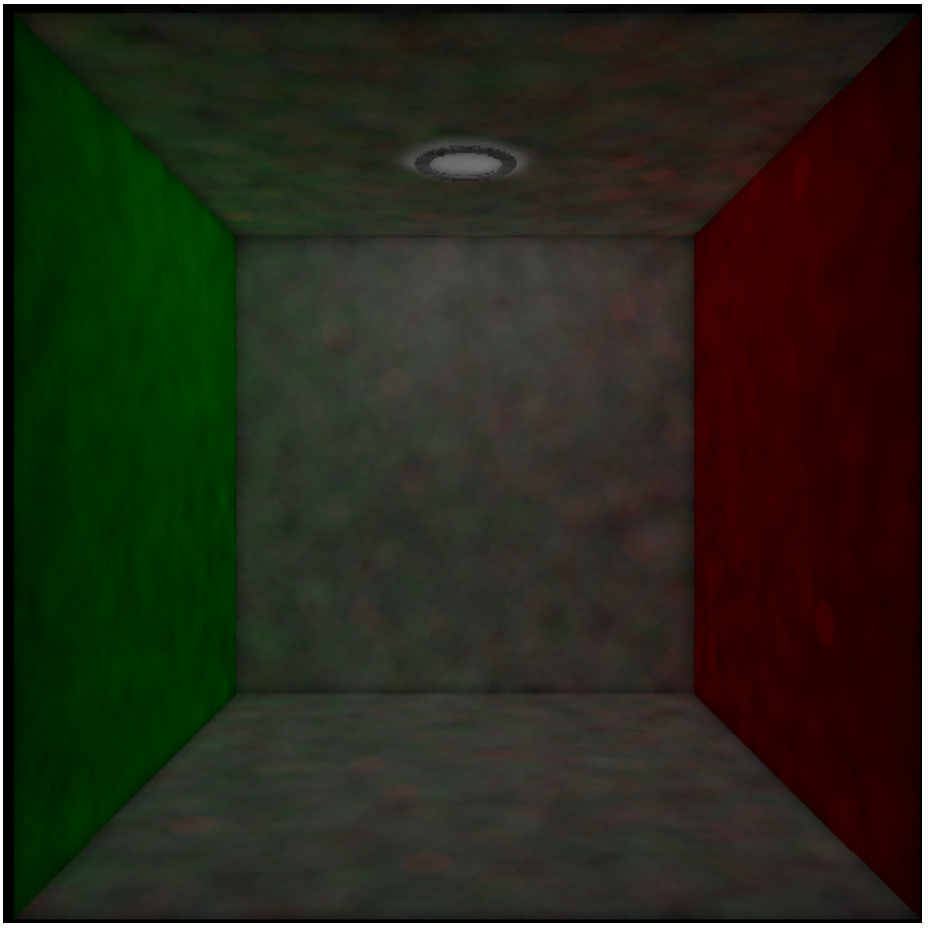


Figure C.2: PPM after four pass with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

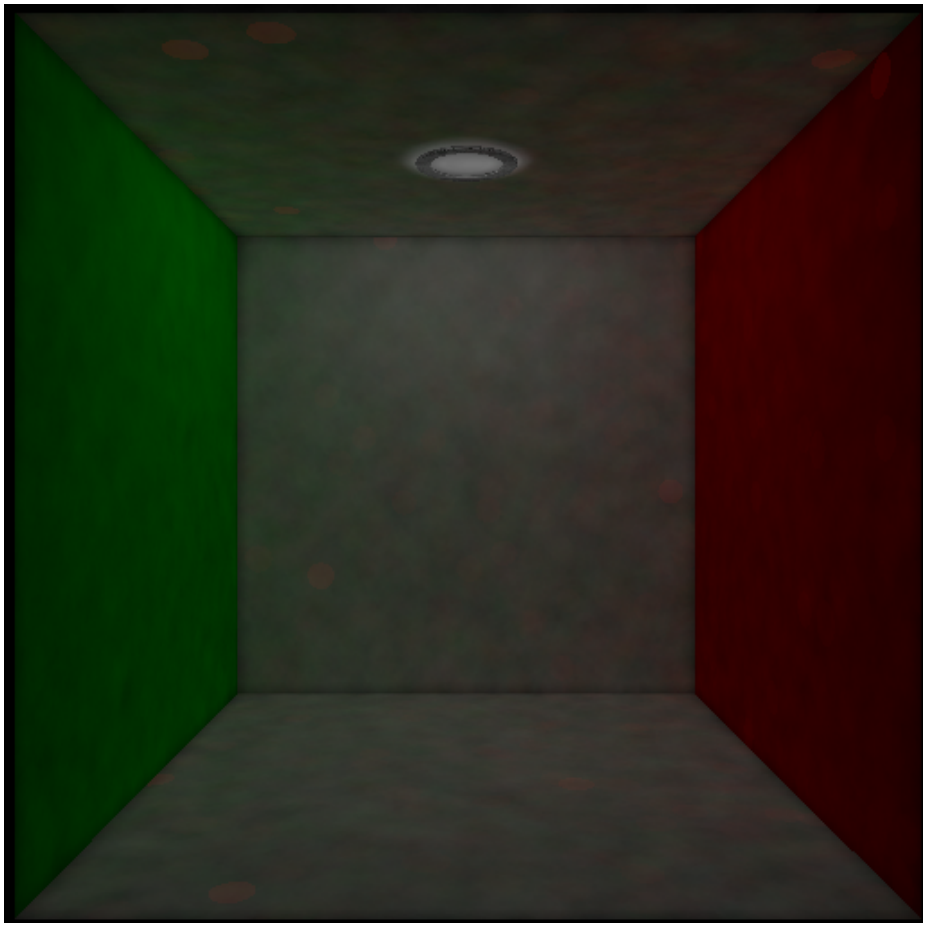


Figure C.3: PPM after 16 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

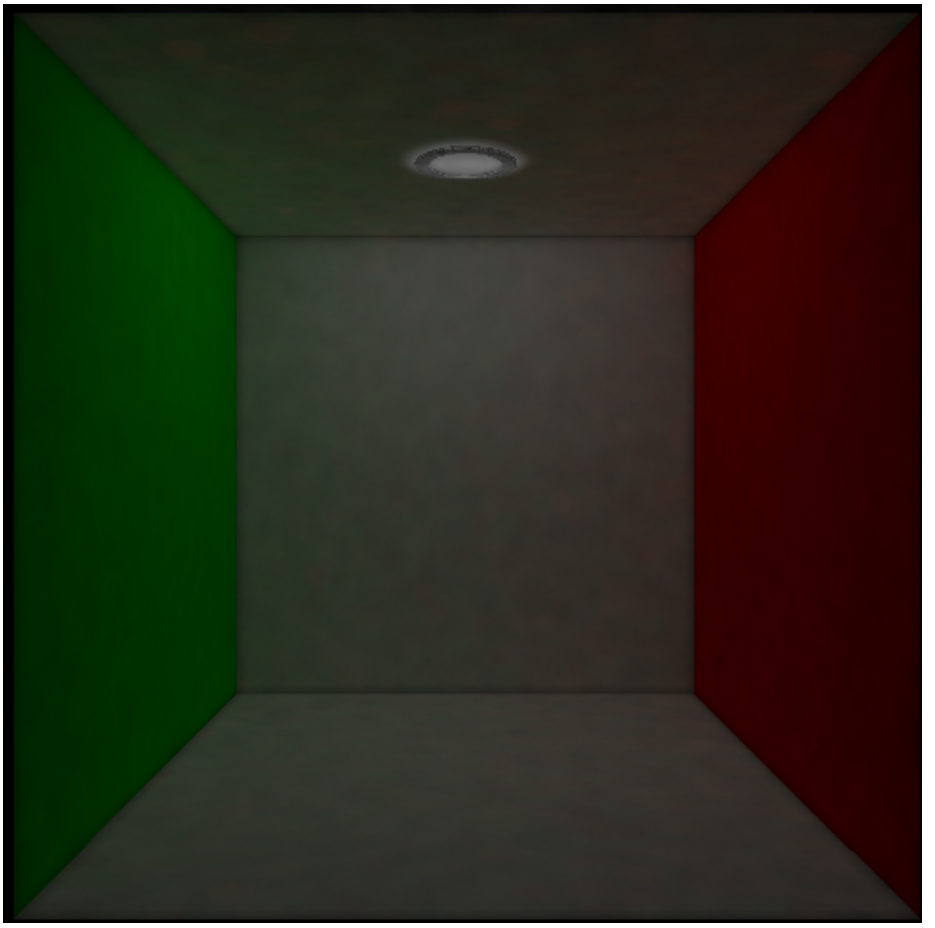


Figure C.4: PPM after 64 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

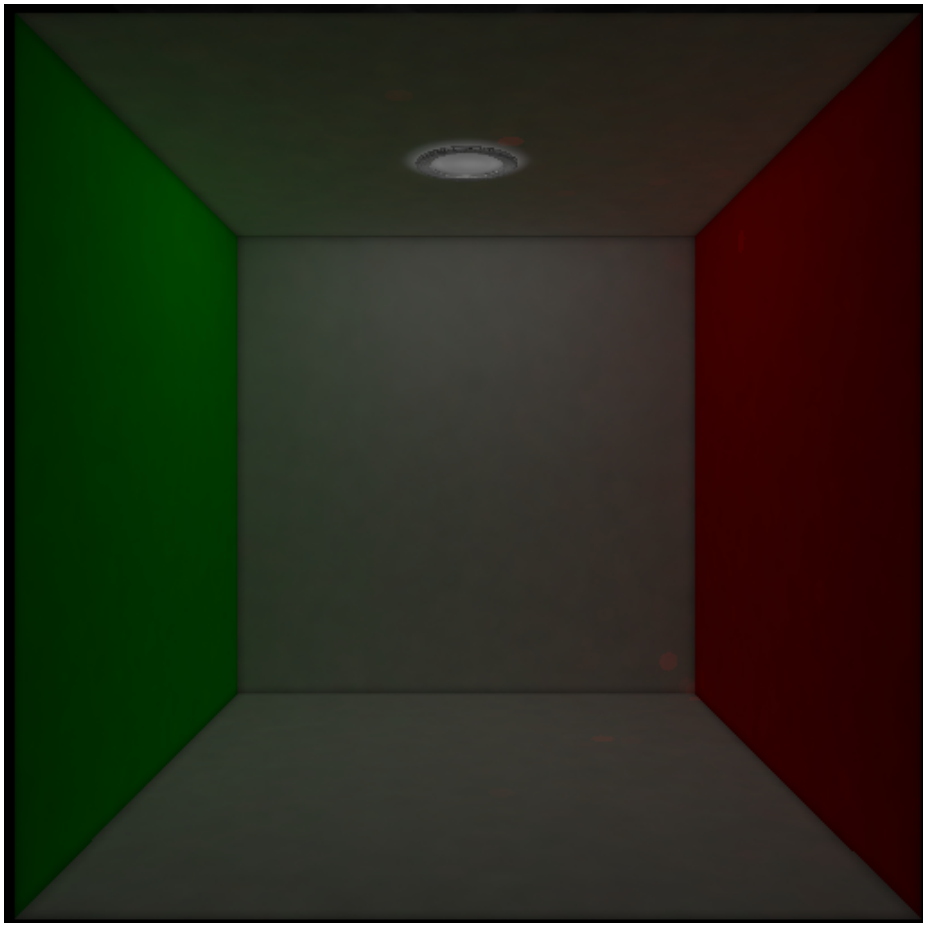


Figure C.5: PPM after 256 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

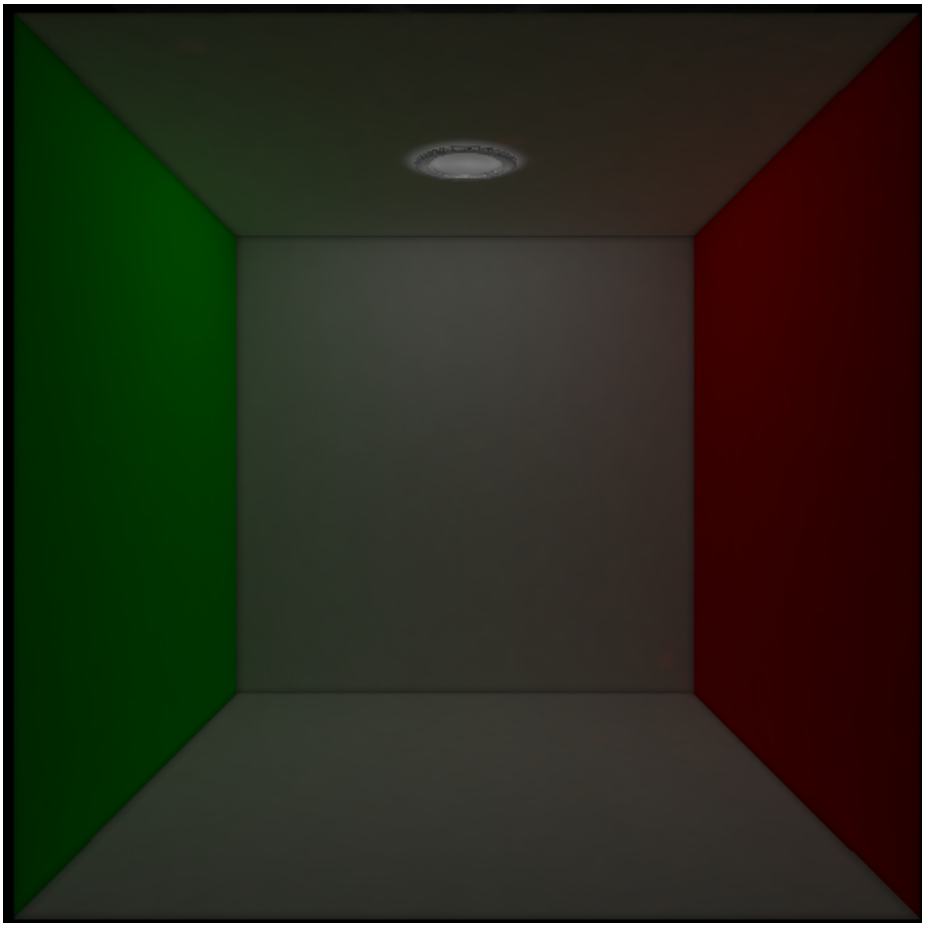


Figure C.6: PPM after 1024 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

Appendix D

SPPM original results

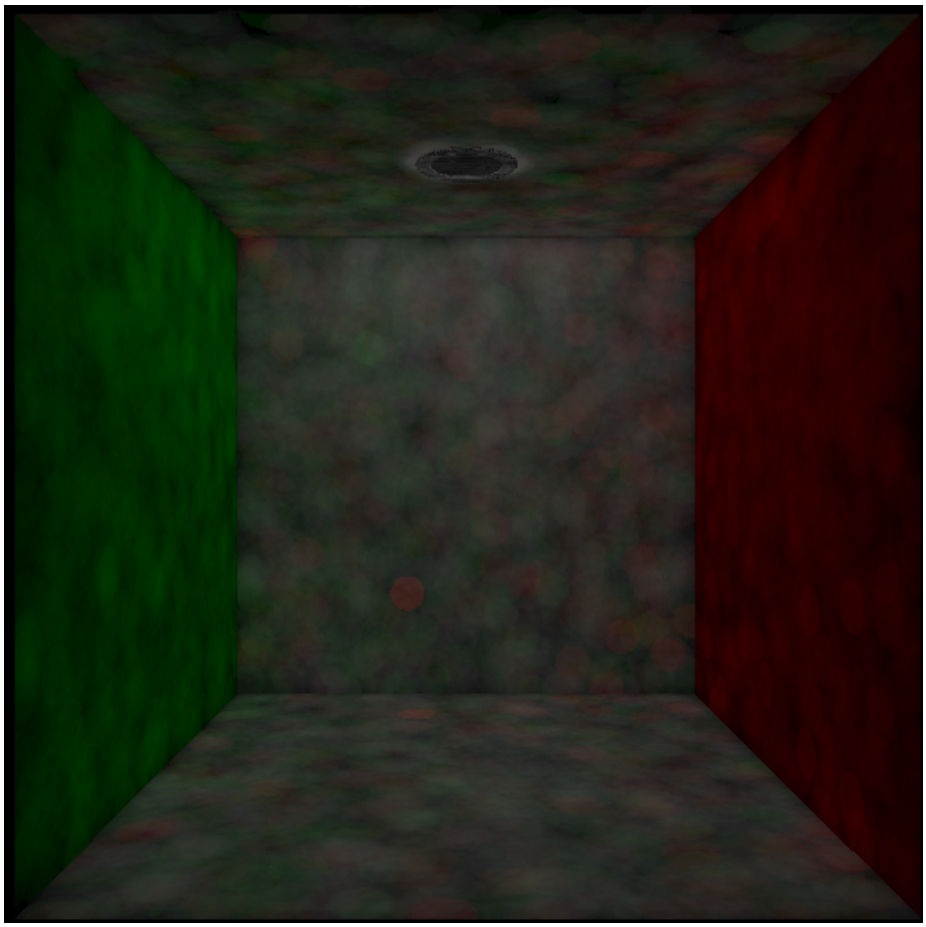


Figure D.1: SPPM after one pass with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8. The normal the the diffuser is turning the wrong way here in the first pass, that is why it is so dark.

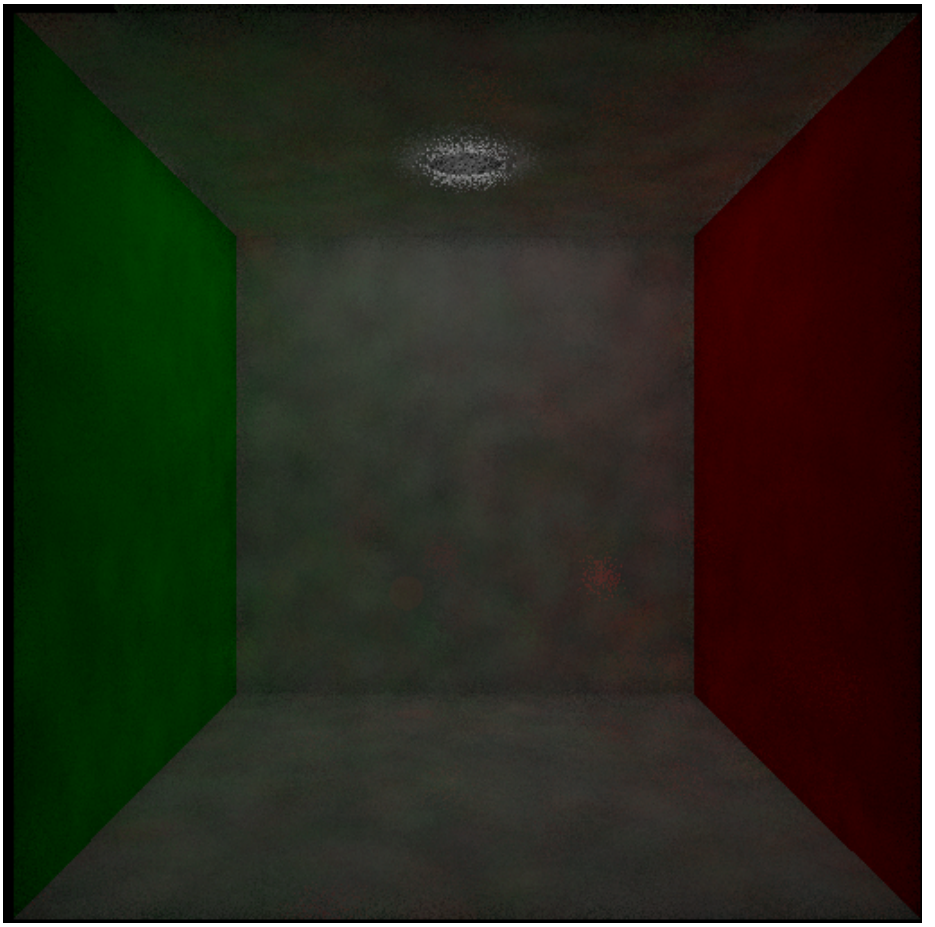


Figure D.2: SPPM after four pass with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

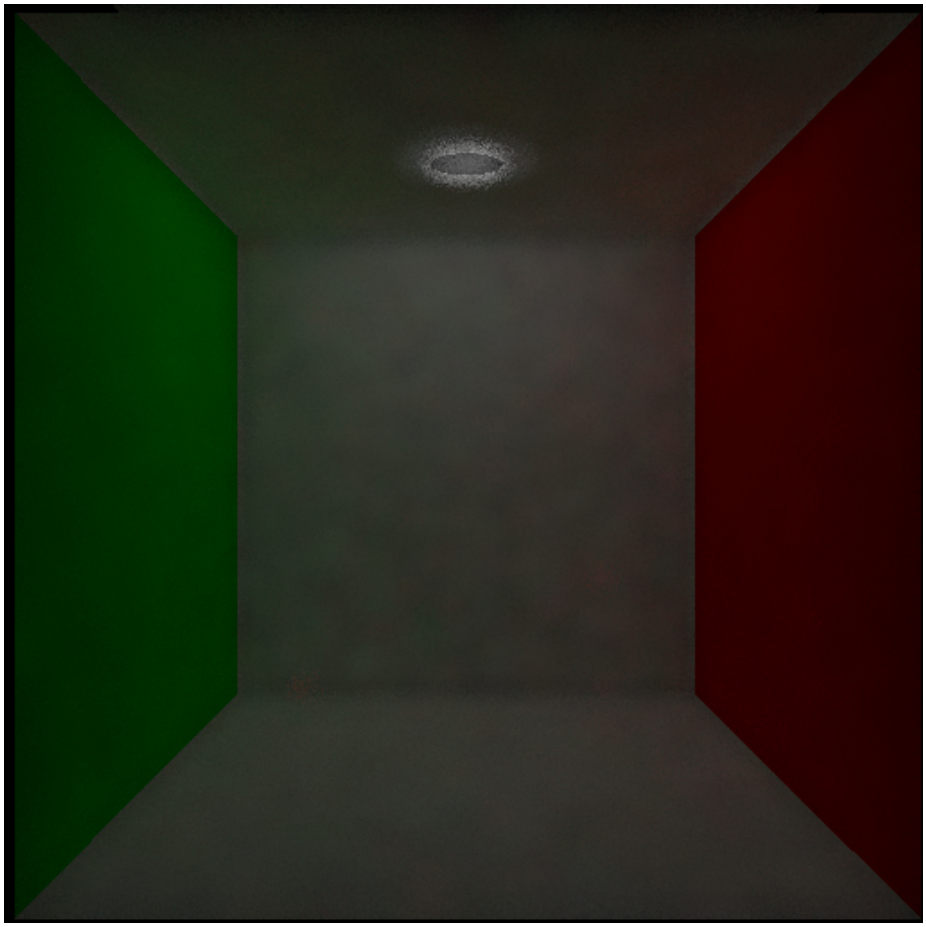


Figure D.3: SPPM after 16 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

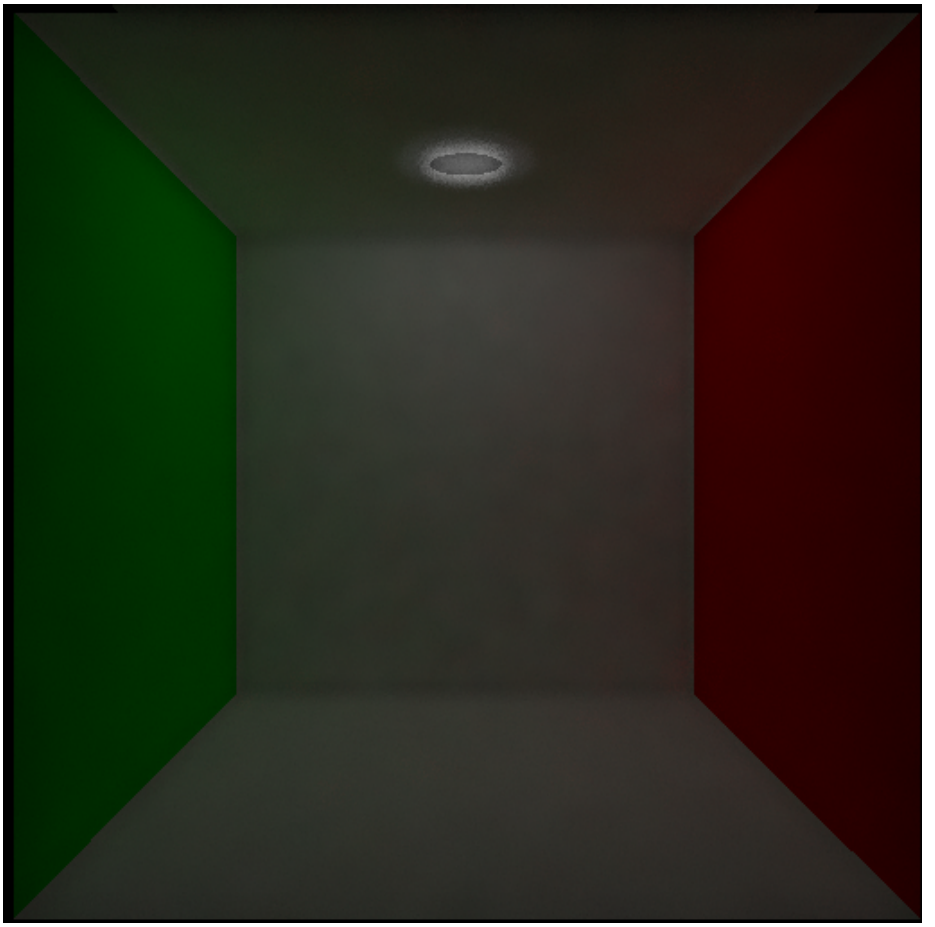


Figure D.4: SPPM after 64 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

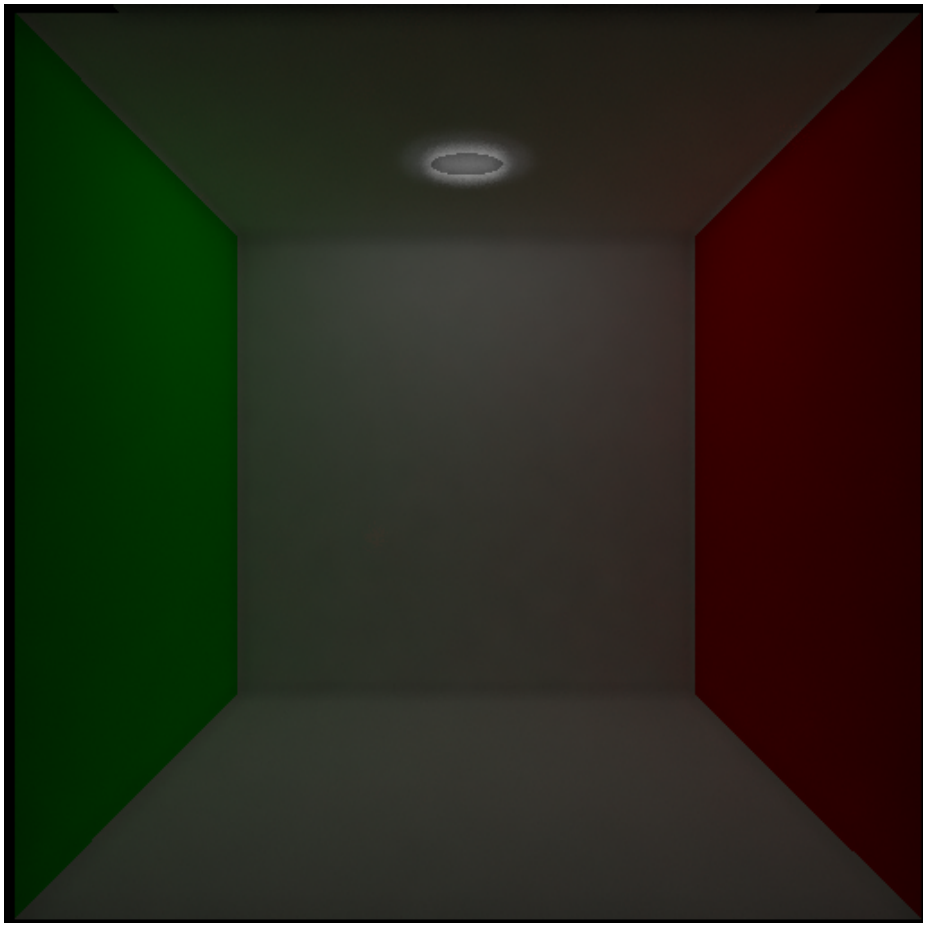


Figure D.5: SPPM after 256 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

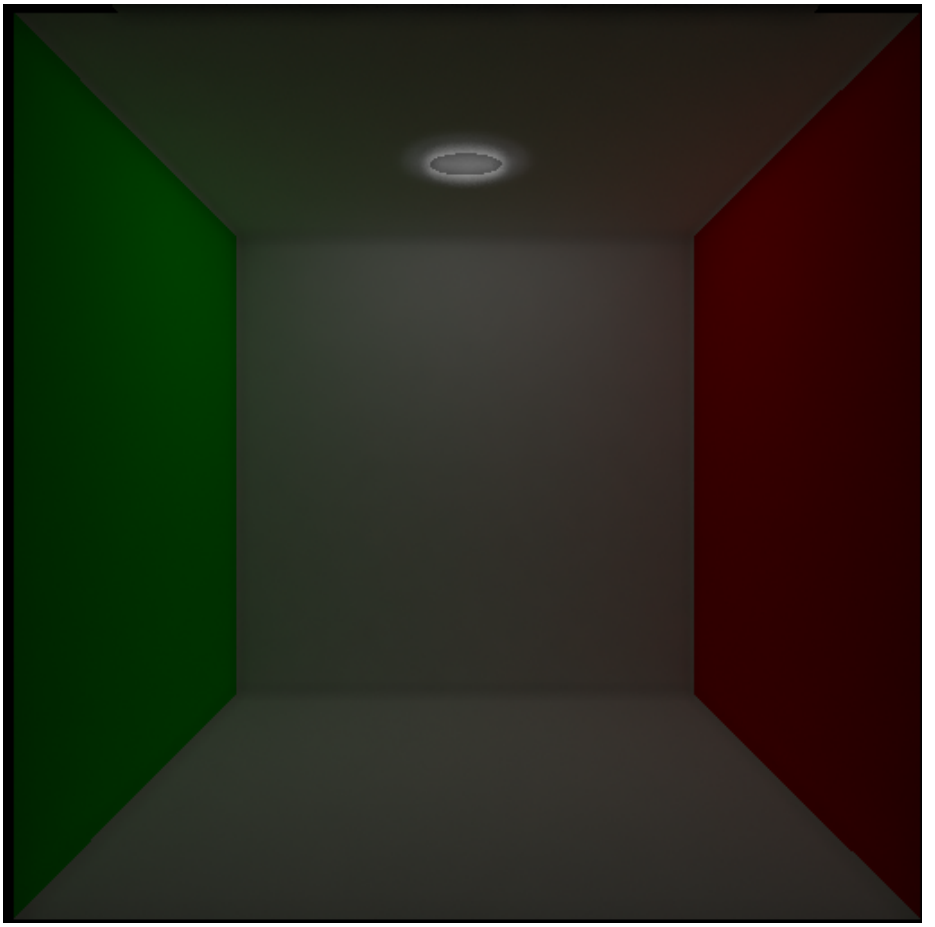


Figure D.6: SPPM after 1024 passes with 0.1 million photons in each pass. The initial radius is 0.3 all over the scene, and the weight is 0.8.

Bibliography

[AZARS06] S. Ahmed, A. Zain-Ahmed, S. Abdul Rahman, and M. H. Sharif. Predictive tools for evaluating daylighting performance of light pipes. 1(4):315–328, 2006.

[DK02] S. Darula and R. Kittler. Cie general sky standard defining luminance distributions. *Proceedings eSim*, pages 11–13, 2002.

[DKK10] Stanislav Darula, Richard Kittler, and Miroslav Kocifaj. Luminous effectiveness of tubular light-guides in tropics. *Applied Energy*, 87(11):3460 – 3466, 2010.

[DS07] S. Dutton and L. Shao. Raytracing simulation for predicting light pipe transmittance. *International Journal of Low-Carbon Technologies*, 2(4):339, 2007.

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Trans. Graph.*, 28:141:1–141:8, December 2009.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 130:1–130:8, New York, NY, USA, 2008. ACM.

[Jen96] H.W. Jensen. Global illumination using photon maps. *Rendering Techniques*, 96:21–30, 1996. `http://www.cs.berkeley.edu/~sequin/CS184/TOPICS/GlobalIllumination/Jensen_egwr96.pdf`.

[Jen01] H.W. Jensen. *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.

[JMK05] D. Jenkins, T. Muneer, and J. Kubie. A design tool for predicting the performances of light pipes. *Energy and buildings*, 37(5):485–492, 2005.

[Kaj86] J.T. Kajiya. The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4):143–150, 1986.

[KDK08] M. Kocifaj, S. Darula, and R. Kittler. Holigilm: Hollow light guide interior illumination method-an analytic calculation approach for cylindrical light-tubes. *Solar Energy*, 82(3):247–259, 2008.

[ORS00] G Oakley, S.B Riffat, and L Shao. Daylight performance of light-pipes. *Solar Energy*, 69(2):89 – 98, 2000.

[PCC07] M. Paroncini, B. Calcagni, and F. Corvaro. Monitoring of a light-pipe system. *Solar Energy*, 81(9):1180–1186, 2007. <ce:title>CISBAT 2005</ce:title>.

[PH10] M. Pharr and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, second edition edition, 2010.

[SS95] PD Swift and GB Smith. Cylindrical mirror light pipes. *Solar Energy Materials and Solar Cells*, 36(2):159–168, 1995.

[VG97] E. Veach and L.J. Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997.

[weba] `http://www.velux.com/SiteCollectionImages/280/Suntunnel_101396-01_w280.jpg`. Last checked 09-01-2012.

[webb] `http://www.velux.com/SiteCollectionImages/280/Suntunnel_101397-01_w280.jpg`. Last checked 09-01-2012.

[webc] `http://geonames.nga.mil/ggmagaz/`. National Geospatial-intelligence Argency - Last checked 09-01-2012.

[webd] `http://aa.usno.navy.mil/data/docs/AltAz.php`. Astronomical Applications Dept. U.S. Naval Obervatory - Last checked 09-01-2012.

[webe] `http://sketchup.google.com/3dwarehouse/details?mid=52b36b12e5b3bce9d0eb3db24373b648`. Last checked 09-01-2012.