Veach——Monte Carlo Path Tracing

For CG course 2023 zju

The following contents are independently completed by myself.

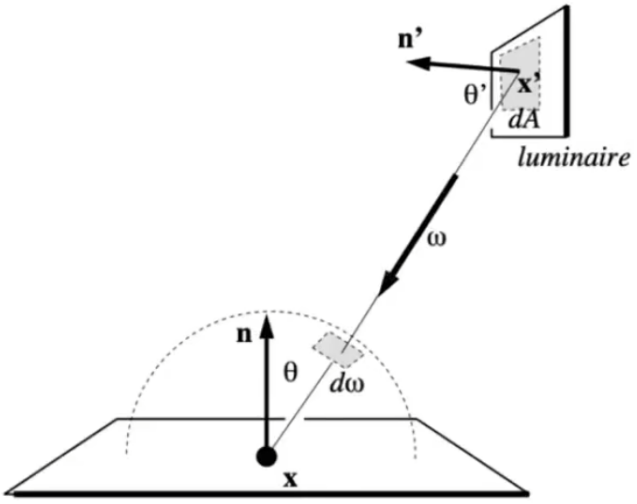
1. Rendering Equation

1. From the perspective of incident ray ω_i

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)(n\cdot\omega_i)d\omega_i \tag{1}$$

2. From the perspective of sampling points $\mathbf{x}'$ on area light

$$\begin{aligned} L_o(x,\omega_o) &= \int_{\Omega^+} L_i(x,\omega_i) f_r(x,\omega_i,\omega_o) cos\theta d\omega_i \\ &= \int_A L_e(\mathbf{x}'\to\mathbf{x}) f_r(x,\omega_i,\omega_o)\frac{cos\theta cos\theta'}{\|\mathbf{x}'-\mathbf{x}\|^2}dA \end{aligned} \tag{2}$$



The above images and formulas partially reference from Games101 course [1]

2. BRDF of Phong Model

$$\begin{aligned} f_r(\mathbf{x},\omega_i,\omega_r) &= f_{r,d}(\mathbf{x},\omega_i,\omega_r) + f_{r,s}(\mathbf{x},\omega_i,\omega_r) \\ &= \rho_d\frac{1}{\pi} + \rho_s\frac{n+1}{2\pi}\cos^n\alpha \end{aligned} \tag{3}$$

(3) references from [2]

- ρ_d is the *diffuse reflectivity*, ρ_s is the *specular reflectivity*.
- α is the angle between incident light ray ω_i and perfect specular reflective direction R of outgoing light ray ω_r .
- n is the Phong exponent(shiness).

> **Note**
>
> Assume θ is the angle between ω_i and surface normal vector $\mathbf{n}$. In (3), if $\theta > \frac{\pi}{2}$, then the first term is 0. If $\alpha > \frac{\pi}{2}$, the second term is 0.

3. Sampling Strategies:

While evaluating equation (2) by Monte Carlo Integration Algorithm, it is necessary to sample on the semi-spherical domain Ω^+. For the purpose of reducing variance, the sampling p.d.f should resemble the integrand as much as possible(eg. one is proportional to another). The integrand in equation (2) is product of three parts: light source term $L_i(x,\omega_i)$, brdf $f_r(x,\omega_i,\omega_o)$, geometric term $cos\theta$ (or $\frac{cos\theta cos\theta'}{\|x'-x^2\|}$). So following sampling techniques take combinations of light source, brdf, geometric term as p.d.f respectively.

A.Sampling the light source

Two different sampling techniques:

1. Uniformly sampling the light source [3]: preferring light source with higher $Radiance$ and then with larger area.

   Steps:

   - Taking corresponding $Radiance$ as probability weights, randomly choose one class of light triangles.
   - Taking its area as probability weights, randomly choose one among those triangles.
   - Uniformly choose one point inside the triangle, and the point is sampling result. Details: First, uniformly sample one coordinate in 2-dimensional region $\{(\beta,\gamma)|\beta\geq 0,\gamma\geq 0,\beta+\gamma\leq 1\}$. Then affinely transform it to light triangle in the space. Usage of *Barycentric Coordinates* (α,β,γ) references from slides [4].

   Alternative: Assuming p.d.f $p_0(\mathbf{x}')\propto L_e(\mathbf{x}'\rightarrow\mathbf{x}),\ \ \mathbf{x}'\in\cup A_i$, A_i is the point set of i^{th} light triangle. Taking the product of $Radiance$ and respective total triangle area as probability weights, sample a $Radiance$. Among the triangles with chosen $Radiance$, sample one while taking area as probability weights. To be continued [5].

2. **Sampling Spherical Triangle** :

   Consider light source term and partial geometry term. Assume $p_1(\mathbf{x}')\propto L_e(\mathbf{x}'\rightarrow\mathbf{x})\frac{cos\theta'}{\|\mathbf{x}'-\mathbf{x}\|^2},\mathbf{x}'\in\cup A_i$, A_i is the point set of i^{th} light triangle.

   With substitution of integral variable, the above is equivalent to $p_2(\omega_i)\propto L_e(\mathbf{x}'\rightarrow\mathbf{x}),\ \ w_i\in\cup A_i'$ [6] , A_i' is i^{th} light triangle's radially projected spherical triangle on the positive hemispherical surface, taking no account of visibility issue.

   Spherical triangles are obtained by projecting light triangles onto hemisphere. Uniformly sample by area among those shperical triangles. Sampling details reference from section 5.2.1 of Arvo1995 [7].

   Specific steps for sampling $p_2(\omega_i)$:

   - $H^2_{\mathbf{x},\mathbf{n}}$ is the positive hemispherical surface with current shading point $\mathbf{x}$ as origin and normal vector $\mathbf{n}$ as central axis. Calculate all radially projected spherical triangles on $H^2_{\mathbf{x},\mathbf{n}}$. The results are completely depicted by unit vectors $\mathbf{A},\mathbf{B},\mathbf{C}$. Calculate intermediate variables $a,b,c,\alpha,\beta,\gamma$,etc. Obtain the area $\mathcal{A}$ of each spherical triangle then.
   - Taking the product of $Radiance$ and area of each projected spherical triangle $L*\mathcal{A}$ as probability weight, sample one spherical triangle among them.
   - Uniformly sample a direction by area from the triangle. It's the sampling result.

     > ⚠ **Warning**
     >
     > The sampled light rays may point to the negative half space defined by normal vector $\mathbf{n}$.

   Further consideration about assuming $p_3(\omega_i)\propto L_e(\mathbf{x}'\rightarrow\mathbf{x})cos\theta,\ \ \omega_i\in\cup A_i'$. To be continued [5].

3. The above two sampling techniques both multiply $Radiance$ and area of light triangles for probability weights. Correctness is demonstrated here.

   Assuming $p(\omega) \propto L, \omega \in \Omega, \Omega = \cup\Omega_i$, L is constant on each Ω_i.

   Then $p(\omega) = cL$, according to probability normalization, $c\int_\Omega Ld\omega = 1$, $c\sum_k \int_{\Omega_k} L_k d\omega = 1$, $c = \frac{1}{\sum_k L_k \int_{\Omega_k} d\omega}$

   So that $P(\omega \in \Omega_i) = c\int_{\Omega_i} L_i d\omega = \frac{L_i \int_{\Omega_i} d\omega}{\sum_k L_k \int_{\Omega_k} d\omega}$, and $\int_{\Omega_i} d\omega$ is exactly the area of (Spherical) light triangle in the current context. ■

B.Sampling the brdf

1. **Phong Model**

   Consider brdf and geometric term. Assuming $p_4(\omega_i) \propto f_r(\mathbf{x}, \omega_i, \omega_r)cos\theta = \rho_d \frac{1}{\pi} cos\theta + \rho_s \frac{n+1}{2\pi}\cos^n \alpha cos\theta$， $\omega_i \in H^2_{\mathbf{x},\mathbf{n}}$, $H^2_{\mathbf{x},\mathbf{n}}$ is the positive hemispherical surface with current shading point $\mathbf{x}$ as origin and normal vector $\mathbf{n}$ as central axis. As α is the angle subtended by perfect specular reflective direction $\mathbf{R}$ of ω_r and incident ray ω_i , accurate integration of $p_4(\omega_i)$ over $H^2_{\mathbf{x},\mathbf{n}}$ seems tedious, even requiring numerical method for inverse element calculation [8].

   Therefore an alternative is adopted: The $cos\theta$ of specular part is discarded. Assuming $p_5(\omega_i) \propto \rho_d \frac{1}{\pi} cos\theta + \rho_s \frac{n+1}{2\pi}\cos^n \alpha$, $\omega_i \in H^2_{\mathbf{x},\mathbf{n}}$, its integral evaluation is still complicated.

   Hence the space of ω_i is expanded. Let $\omega_i \in H^2_{\mathbf{x},\mathbf{n}} \cup H^2_{\mathbf{x},\mathbf{R}}$, then $p_6(\omega_i) = c(\rho_d \frac{1}{\pi} cos\theta + \rho_s \frac{n+1}{2\pi}\cos^n \alpha)$, $\omega_i \in H^2_{\mathbf{x},\mathbf{n}} \cup H^2_{\mathbf{x},\mathbf{R}}$. $H^2_{\mathbf{x},\mathbf{R}}$ is the positive hemispherical surface with current shading point $\mathbf{x}$ as origin and $\mathbf{R}$ as central axis.

   Integrate over $p_6(\omega_i)$:

$$1 = \int_{H^2_{\mathbf{x},\mathbf{n}} \cup H^2_{\mathbf{x},\mathbf{R}}} c(\rho_d \frac{1}{\pi} cos\theta + \rho_s \frac{n+1}{2\pi}\cos^n \alpha) d\omega_i \tag{4}$$

$$1 = c(\rho_d \int_{H^2_{\mathbf{x},\mathbf{n}}} \frac{1}{\pi} cos\theta d\omega_i + \rho_s \int_{H^2_{\mathbf{x},\mathbf{R}}} \frac{n+1}{2\pi}\cos^n \alpha d\omega_i) \tag{5}$$

$$1 = c(\rho_d + \rho_s) \tag{6}$$

$$c = \frac{1}{\rho_d + \rho_s}$$

   In order to deduce from (4) to (5), domain of integration could be separated into 1 intersection and 2 difference sets and evaluate integrand over them. Provement of normalization from (5) to (6) references from course slides [9].

   Steps for sampling $p_6(\omega_i)$ follow:

   - Taking ρ_d and ρ_s as probability weights, sample a random to choose diffuse part or specular part.
   - If diffuse part chosen, sample a direction as result from cosine weighted hemisphere $H^2_{\mathbf{x},\mathbf{n}}$.
   - If specular part chosen, sample a direction as result from Phong lobe $H^2_{\mathbf{x},\mathbf{R}}$.

     > **Note**
     >
     > Do not forget transforming $\mathbf{R}$ based coordinates to $\mathbf{n}$ based coordinates in world coordinate system.

   Details of sampling from cosine weighted hemisphere and Phong lobe reference from course slides [9].

   > **Warning**
   >
   > Sampled direction may point to negative half space defined by normal vector $\mathbf{n}$.

2. Blinn-Phong Model(To be continued [5])

   May be easier, because α in $p_4(\omega_i)$ is replaced by θ.

4. Multiple Importance Sampling(MIS)

A.Multi-sample Model

Assume: Following intergal needs evaluation

$$\int_\Omega f(x)d\mu(x) \tag{7}$$

and the domain of integration Ω, function $f:\Omega\to\mathbb{R}$, measurement μ are all given。

- A set of n different sampling techniques on the domain Ω is given, whose corresponding density functions are labeled $p_1, p_2 \dots p_n$.
- Given any point $x\in\Omega$, $p_i(x)$ can be evaluated.

B.Multi-sample estimator

$$F=\sum_{i=1}^{n}\frac{1}{n_i}\sum_{j=1}^{n_i}w_i(X_{i,j})\frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{8}$$

F is an estimator of equation (7), among which $X_{i,j}$ is j^{th} sample from i^{th} sampling technique, for $j=1,\dots,n_i$. w_i is weighting function of i^{th} sampling technique, which satisfies the following two conditions:

- $\sum_{i=1}^{n}w_i(x)=1$ whenever $f(x)\neq 0$, and
- $w_i(x)=0$ whenever $p_i(x)=0$.

Therefore F is a MonteCarlo estimator of equation (9), in which i^{th} sampling technique contributes to i^{th} summation term.

$$\int_\Omega f(x)d\mu(x)=\sum_{i=1}^{n}\int_\Omega w_i(x)f(x)d\mu(x) \tag{9}$$

It's provable that F is unbiased.

C.Balance heuristic

$$\hat{w}_i(x)=\frac{n_ip_i(x)}{\sum_k n_kp_k(x)} \tag{10}$$

It can be proved that estimator with equation (10) being weighting function guarantees lower variance upper bound.

Above basic concepts about MIS reference from Stanford CS348b course slides [10].

D.Example here

In this project, reflectance equation is the integral to be evaluated. Generally there are two sampling strategies, each with sample size $n_i=1$:

- Sample the light source, with p.d.f p_0, then sampled direction X_0 is obtained.
- Sample the brdf, with p.d.f p_1, then sampled direction X_1 is obtained.

According to equation $(7)\sim(10)$, the estimator of $L_o(x,w_o)$ could be:

$$F=\frac{f(X_0)}{p_0(X_0)+p_1(X_0)}+\frac{f(X_1)}{p_0(X_1)+p_1(X_1)} \tag{11}$$

in which $f(\omega) = L_i(\mathbf{x}, \omega) f_r(\mathbf{x}, \omega, \omega_o) cos\theta$

Further attempts could be made about power heuristic [10].(To be continued [5])

5. Russian Roulette

The Russian Roulette parameters in this project are configured as follow:

P_RR: The probability that light ray continue to exit current shading point $\mathbf{x}$ and calculate $Radiance$ recursively.

Sampling the light source + Sampling the brdf: P_RR = 0.6

Only Sampling the brdf: P_RR = 0.6

6. Ray Intersection and Accelerating Structure

Ray Intersection: Intersection with triangle, details reference from course slides [4].

Accelerating structure: 3d uniform grid.

Berkeley CS184 [11] mentioned that the heuristic for grid cell number is #cells = 27 * #objs. There are 3092 facets in the Veach scene. So the number of grid cells is set to be $n_0 = 100000$.

3D DDA is adopted instead of Bresenham. The core algorithm is checking whether the next integer boundary of grid has been reached. Then convert 3d case to 2d case , 2d case to 1d case.

> **Note**
>
> Be careful about non-integer starting point.

7. Core Algorithm

There are 3 different procedures of calculating exiting $Radiance$ of shading point $\mathbf{x}$. Each subprogram could be invoked individually by the rendering process in `main.cpp`.

1. Direct and Indirect Illumination: Incident $Radiance$ could be classified into **direct** and **indirect** by the source of incident light ray. Calculate them respectively and add up.
    - Evaluate direct illumination by 1 sample from light source.(As in 3.A, either uniformly sampling the light source or sampling spherical triangle)
    - Evaluate indirect illumination by 1 sample from brdf.
    - If the sampled direction from brdf points to light triangles, the resulting indirect contribution is set to be 0. Because the $Radiance$ has been included by direct illumination calculation.
    - Light triangles only take account of exiting $Radiance$ caused by self illumination, regardless of reflectance $Radiance$ caused by incident light rays.
    - Ordinary triangles only consider reflectance $Radiance$.

    ```
    //main.cpp
    //Evaluate the exiting Radiance in the direction of 'wo' at the intersection
    'point'.
    RadianceRGB shade(intersec_result point, vec wo);
    ```

    The above core process references from Games101 [12].

The connection with **Multi-sample estimator** :

Lemma 1: The classification about direct and indirect illumination as well as the algorithm, could be deemed as an example of estimator F in equation (8). (Specifics about w_i follow)

- Two kinds of sampling techniques: Sampling the light source with p_0, Sampling the brdf with p_1.
- Assume current shading point $\mathbf{x}$, normal vector $\mathbf{n}$. A' is the point set obtained by radially projecting all light triangles on the positive hemispherical surface $H^2_{\mathbf{x},\mathbf{n}}$, taking no account of visibility issue.
- $B' = \left\{ \mathbf{a} \in A' \,\middle|\, \begin{array}{l} The\ light\ ray\ shooting\ from\ \mathbf{x}\ pointing\ to\ the\ direction\ of\ \mathbf{a}\ can\ intersect \\ with\ any\ light\ triangle\ instead\ of\ being\ blocked\ in\ the\ middle. \end{array} \right\}$
- It can be inffered that $B' \subseteq A'$, and $A - B'$ is the set of points whose path towards light triangle is blocked.
- U is the set of points on positive hemispherical surface $H^2_{\mathbf{x},\mathbf{n}}$.
- The weighting functions of 2 sampling techniques are defined as follows :

$$w_0(\omega_i) = \begin{cases} 1 & \omega_i \in B' \\ 0 & \omega_i \in U - B' \end{cases} \tag{12}$$

$$w_1(\omega_i) = \begin{cases} 0 & \omega_i \in B' \\ 1 & \omega_i \in U - B' \end{cases} \tag{13}$$

It can be verified that function (12) and (13) satisfy the two condition proposed in Multi-sample estimator. ■

2. Brdf Only:
   - Light triangles only take account of exiting $Radiance$ caused by self illumination, regardless of reflectance $Radiance$ caused by incident light rays.
   - Ordinary triangles only consider reflectance $Radiance$.
   - Reflectance $Radiance$ is evaluated by 1 **sample from brdf**.

```
//main.cpp
//Evaluate the exiting Radiance in the direction of 'wo' at the intersection
'point'.
RadianceRGB shade_with_brdf(intersec_result point, vec wo)
```

3. **MIS**:
   - Light triangles only take account of exiting $Radiance$ caused by self illumination, regardless of reflectance $Radiance$ caused by incident light rays.
   - Ordinary triangles only consider reflectance $Radiance$.
   - Obtain 1 sample from **Spherical Triangle Sampling** and 1 sample from **brdf Sampling** . Calculate weighted average of corresponding contribution with Balance heuristic, as shown in equation (11) .

```
//main.cpp
//Evaluate the exiting Radiance in the direction of 'wo' at the intersection
'point'.
RadianceRGB shade_with_mis(intersec_result point, vec wo)
```

> **Note**
>
> Direct and indirect illumination is not distinguished here.

According to Lemma 1, procedure 1 conforms to Multi-sample estimator model. And the Balance heuristic is proved to possess variance with significant lower upper bound when compared with any other weighting heuristics.

Therefore, MIS should own lower variance than former two algorithms.

It will be confirmed in the following experiment results.

8. ToneMapping

Gamma compression is used for ToneMapping.

Maximum of $Radiance$ is set to be: 380 (Only for Veach scene)

γ = 0.25

9. Experimental screenshots

10 ray per pixel

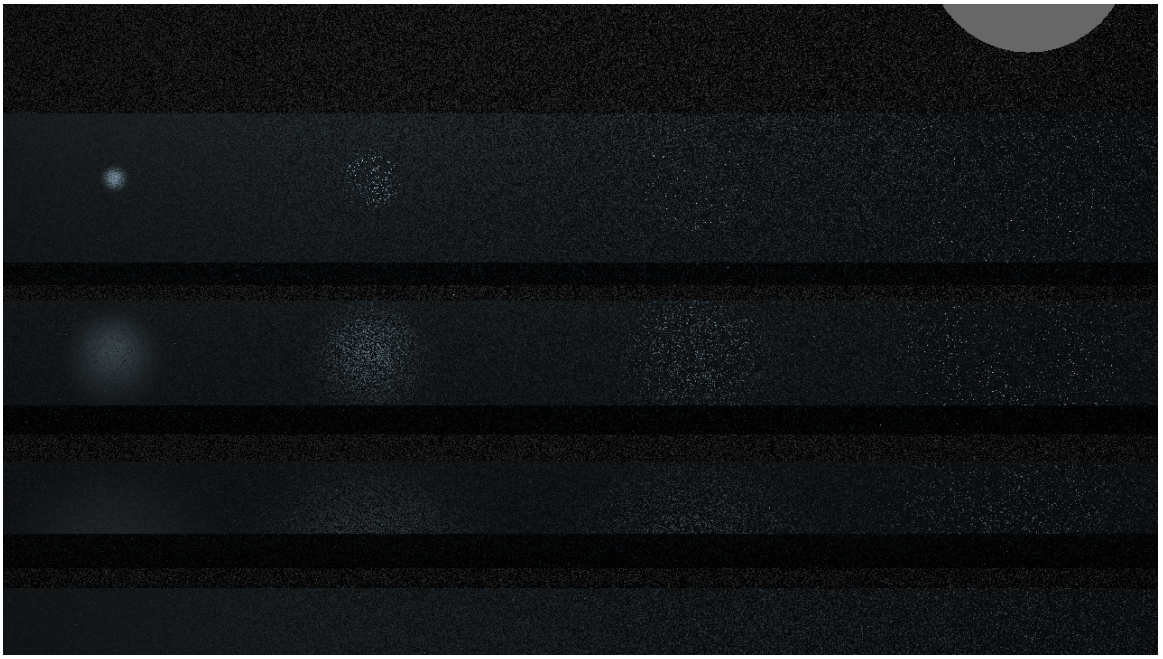
Hyper-parameters configured in xml:

```
<camera type="perspective" width="1280" height="720" fovy="20.1143">
    <eye x="28.2792" y="5.2" z="1.23612e-06"/>
    <lookat x="0.0" y="2.8" z="0.0"/>
    <up x="0.0" y="1.0" z="0.0"/>
</camera>
```

1. Direct and Indirect Illumination

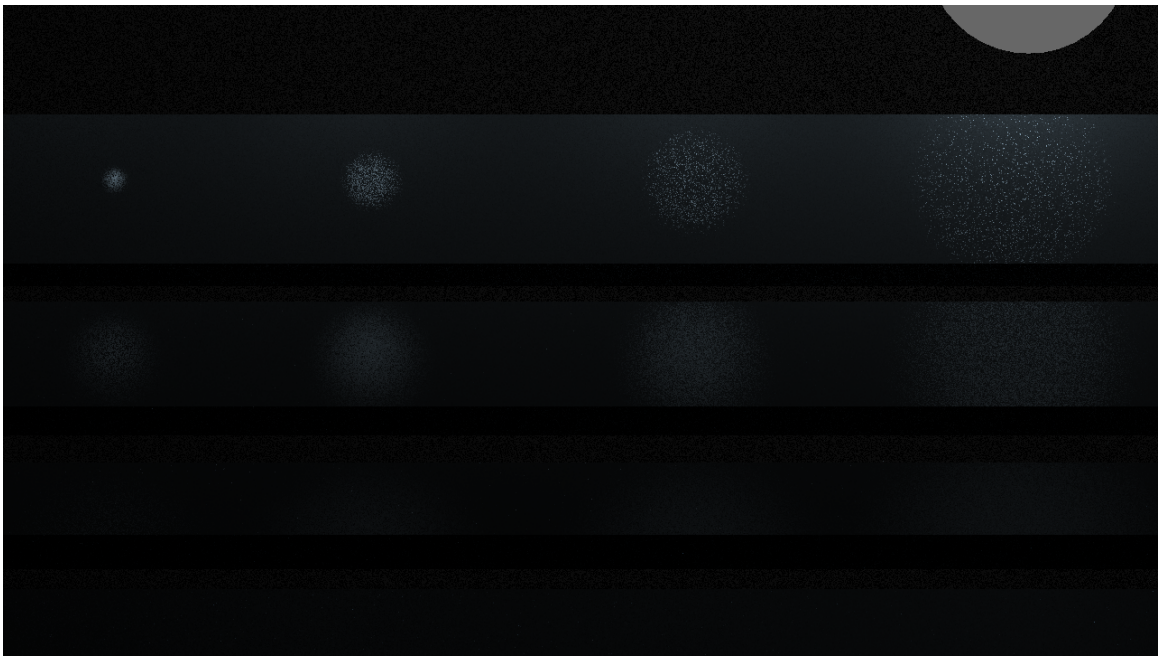
   Uniformly sample the light source, with twice distant from staring point.('eye point' stay twice of original distance away from 'lookat point')

   Rendering time: 330 min.

   Spherical Triangle Sampling, with twice distant from staring point.

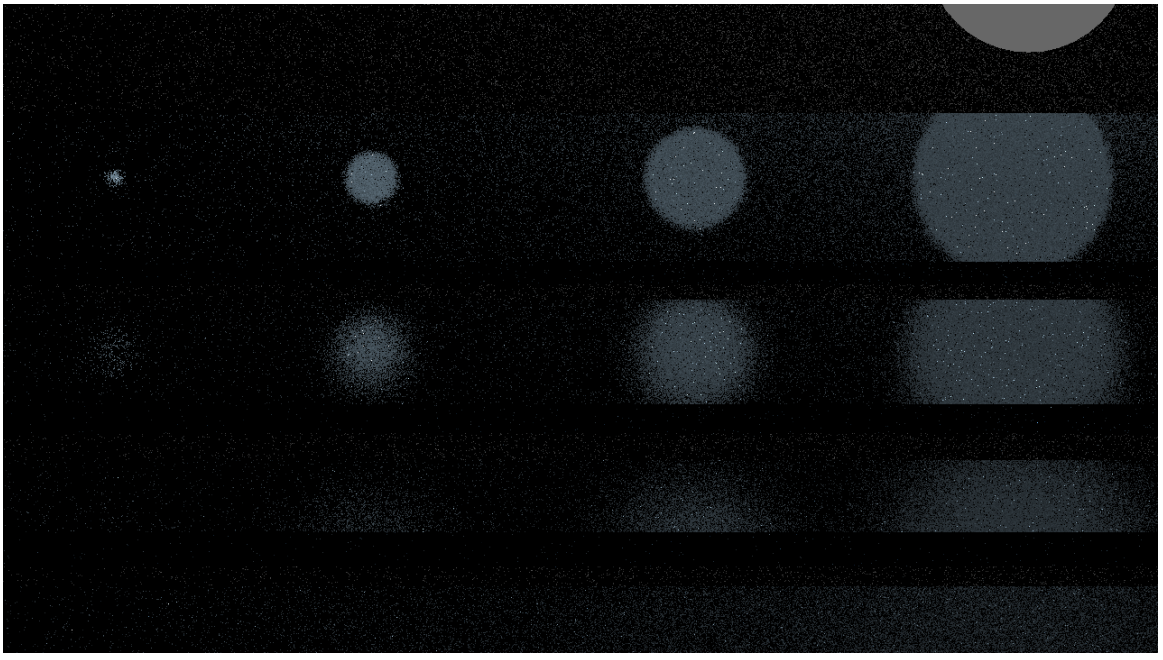
   Rendering time: 1300 min.



2. Brdf Only

   Twice distant from staring point.

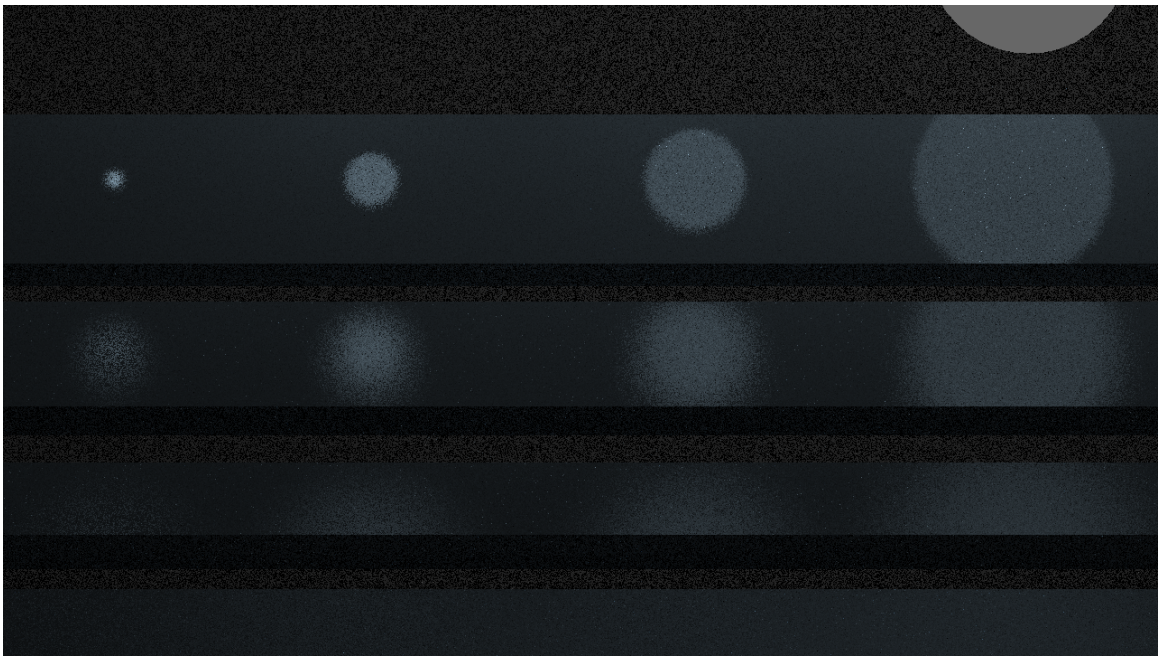
   Rendering time: 60 min.



3. **MIS**

   Twice distant from staring point.

   Rendering time: 1388 min.



Once distant from staring point.(Namely the original parameters assigned by the XML file)

Rendering time: 1125 min.

The background of the following figure lack illumination partially. I debugged many times, but fail to find out. To be continued [5].

The above results show that Spherical Triangle Sampling lead to lower variance, but consumes much more time.

10. Structure of Code Implementation

Project Name: Monte Carlo Path Tracing

Programming Language: ISO C++20

IDE: Visual Studio2019

Dependent third party libraries: tiny_obj_loader(for parsing obj file), pugixml(for parsing xml file), easyx(for visualize framebuffer data in the window)

Git Repository: https://github.com/luotong96/Monte_Carlo_Path_Tracing.git

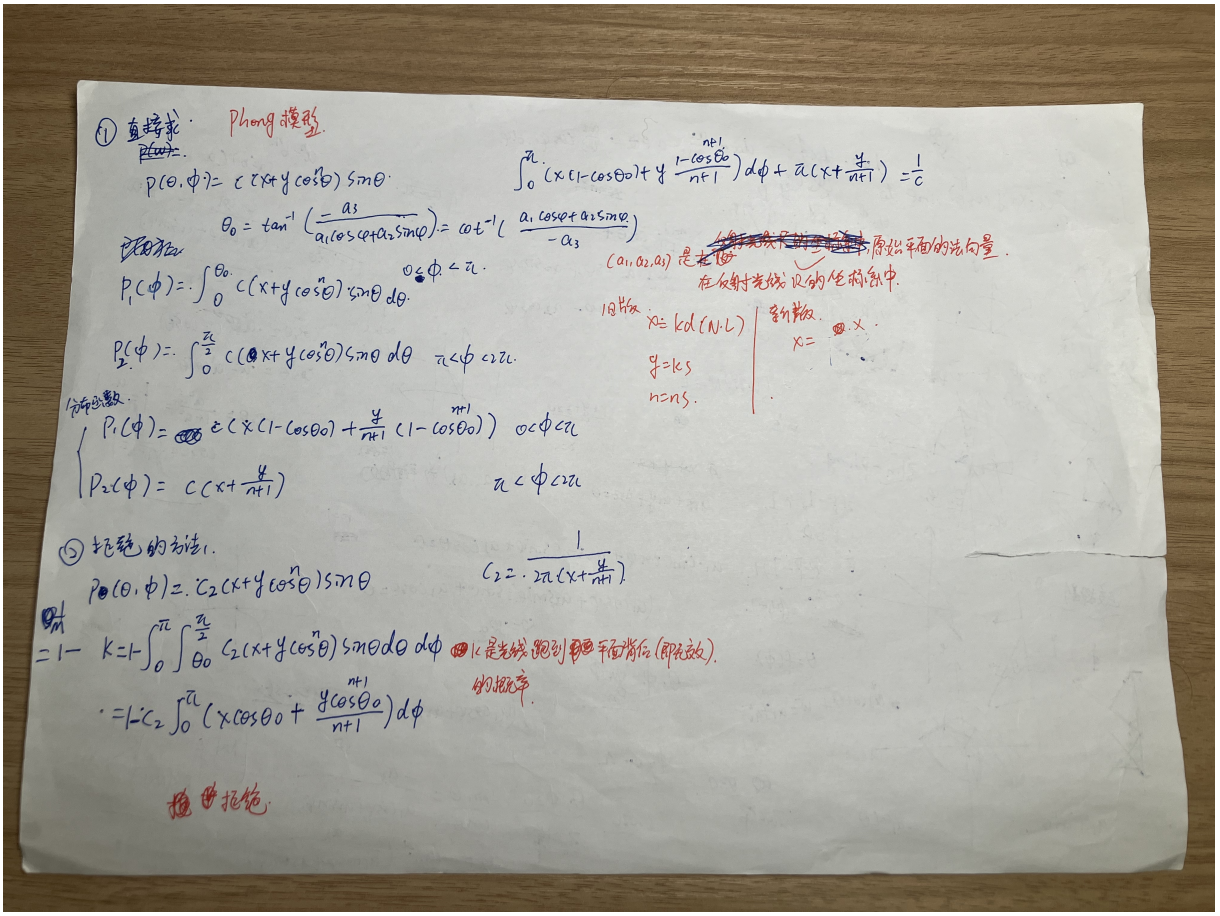
cpp Files	Functional description
BRDF.cpp	Define data type of BRDF. Evaluating BRDF of Phong Model. Sample light ray from Phong Model.
main.cpp	Main rendering process. Hold instances of Myobj , Mylight. Carry out all 3 path tracing procedures by implement *shade* functions. Map screen coordinate system to world coordinate system. Visualize rendering results in window, and save it as images on the local file system.
matrix3d.cpp	Implement 3d matrix operations.
Mylight.cpp	Parse and manage light source data from xml file. Implement uniformly sample the light source. Implement Spherical Triangle Sampling.
Myobj.cpp	Parse obj file and store all data of veach scene with the help of tiny_obj_loader. Implement 3d uniform grid. Implement light ray intersection.
RadianceRGB.cpp	Define data structure of $Radiance$. Implement ToneMapping.
vec.cpp	Implement all 3d vector operations.

Multi-thread: To be continued [5].

Bibliography

1. GAMES101: 现代计算机图形学入门 (ucsb.edu) ↩
2. Assignment 3: Phong and Multiple Importance Sampling (mcgill.ca) ↩
3. Named by myself, may not be precise. ↩
4. 8-ray_casting_2023.pdf ↩ ↩
5. To be continued means it haven't been implemented. ↩ ↩ ↩ ↩ ↩ ↩
6. Unless otherwise specified, treat expression about solid angle $\omega \in PointSet$ as $\omega \in \{\omega_i | \mathbf{x} + \omega_i \in PointSet\}$, $\mathbf{x}$ is the current shading point. ↩
7. Arvo J R. Analytic methods for simulated light transport[D]. Yale University, 1995. ↩

8. Direct integral evaluation, but is not correct. May benefit detailed analysis when sampling the Blinn-Phong brdf.



↩

9. 11-mc_ii_2023.pdf ↩ ↩

10. veach-chapter9.pdf (stanford.edu) ↩ ↩

11. 10-acceleration (berkeley.edu) ↩

12. GAMES101_Lecture_16 (ucsb.edu) ↩