# CSC4140 Assignment 7

Computer Graphics

April 26, 2022

## Ray Tracing II

This assignment is 8% of the total mark.

Strict Due Date: 11:59PM, May $1^{th}$, 2022

Student ID: 119010265

Student Name: Shi Wenlan

This assignment represents my own work in accordance with University regulations.

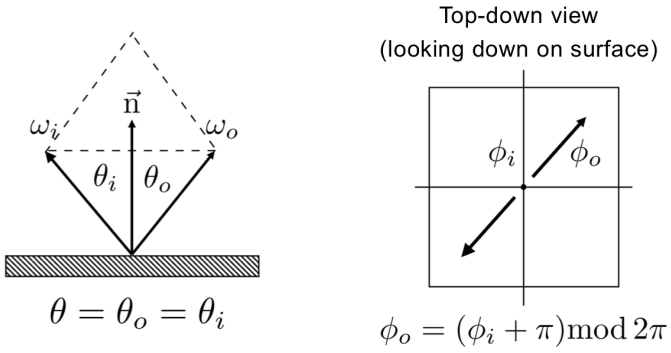
Signature: Shi Wenlan

# 1 Overview

In this project, I learned and implemented glass material, glass material through calculate reflection and refraction, algorithm of using an infinite environment light through uniform sampling or importance sampling, and algorithm to simulate a thin lens to enable the depth of field effect.

# 2 Mirror and Glass Materials

**Idea of implementing reflection and mirror material:** $BSDF :: reflect(...)$ can be implemented according to following formula:




$$\omega_o + \omega_i = 2\cos\theta\,\vec{n} = 2(\omega_i \cdot \vec{n})\vec{n}$$

$$\omega_o = -\omega_i + 2(\omega_i \cdot \vec{n})\vec{n}$$

Figure 1: reflection formula

In the $MirrorBSDF :: sample\_f(...)$, use $BSDF :: reflect(...)$ to sample the unique reflective in-direction $wi$, and set the $pdf$ as 1.0. Note that this function should return $reflectance$ / $abs\_cos\_theta(*wi)$ rather than simply $reflectance$ to cancel out the $abs\_cos\_theta(*wi)$ will be multiplied in function $at\_least\_one\_bounce\_radiance(...)$ since we assume that there all the energy except the energy absorbed by the mirror surface is transmitted to the out-direction.

**Idea of implementing refraction and glass material:** In the function $BSDF :: refract(...)$, if the $z$ attribute of given out-direction is positive, then the ray is entering the non-air material, and $\eta$ equals to 1.0 / ($ior$: index of refraction), otherwise the ray is exiting the non-air material, and $\eta$ equals to $ior$. Then compute $\delta$ through:

$$\delta = 1 - eta^2 * (1 - outdirection.z^2)$$

If $\delta$ is negative, then it is a total reflection, just return false. Otherwise, compute in-direction through following formula and return true:

$$\omega_i.x = -\eta\omega_o.x, \quad \omega_i.y = -\eta\omega_o.y, , \quad \omega_i.z = \mp\sqrt{1-\eta^2(1-\omega_o.z^2)},$$

Figure 2: refraction formula

In the function $GlassBSDF :: sample\_f(...)$, if $BSDF :: refract(...)$ return false, then do as $MirrorBSDF :: sample\_f(...)$ since it is a total reflection, otherwise compute Schlick's reflection coefficient which is used to approximate Fresnel equations model using following formula:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

*where*

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

Figure 3: Schlick's reflection coefficient formula

where $\theta$ is the angle between the direction from which the incident light is coming and the normal of the interface ( = (0,0,1)) between the two media, hence $cos(\theta) = abs\_cos\_theta(outdirection)$. And $n_1$, $n_2$ are the indices of refraction of the two media at the interface and $n_2$ is the reflection coefficient for light incoming parallel to the normal (i.e. the value of the Fresnel term when ( = 0 or minimal reflection). In computer graphics, one of the interfaces is usually air, meaning that $n_1$ very well can be approximated as 1. The program has a probability of $R$ to perform like a mirror reflection with pdf = R and return $R$ * $reflectance$ / $abs\_cos\_theta(*wi)$, and otherwise perform refraction with pdf = (1 - R) and return $(1 - R)$ * $reflectance$ / $abs\_cos\_theta(*wi)$ / $\eta^2$ where $\eta^2$ used to perform the function that radiance concentrates when a ray enters a high index of refraction material (low $\eta$) and disperses when a ray exits such a material (high $\eta$)

**Sequence of six images of scene CBspheres.dae** The rendered images are as following (/dae/sky/CBspheres.dae with 64 samples per pixel and 4 samples per light):

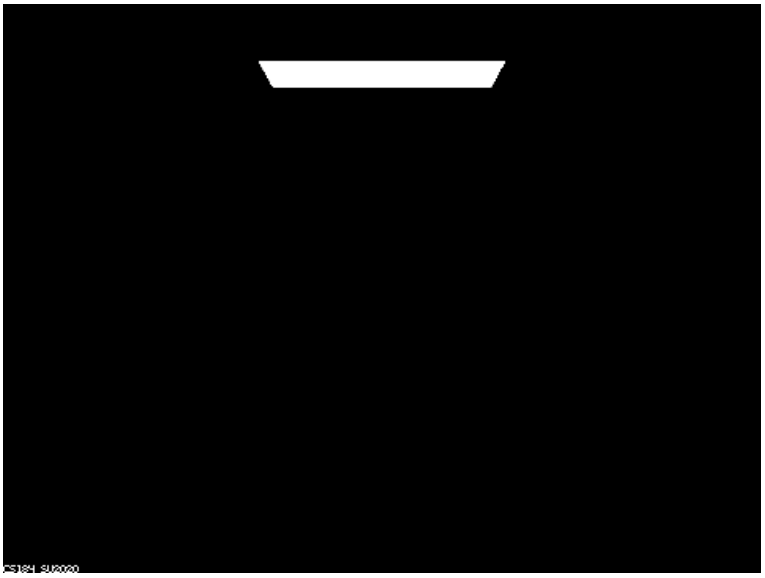


Figure 4: max ray depth = 0

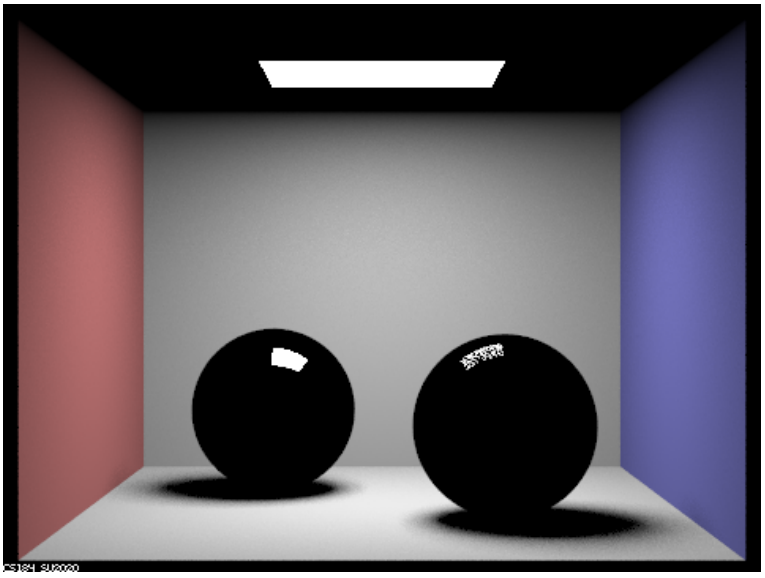


Figure 5: max ray depth = 1

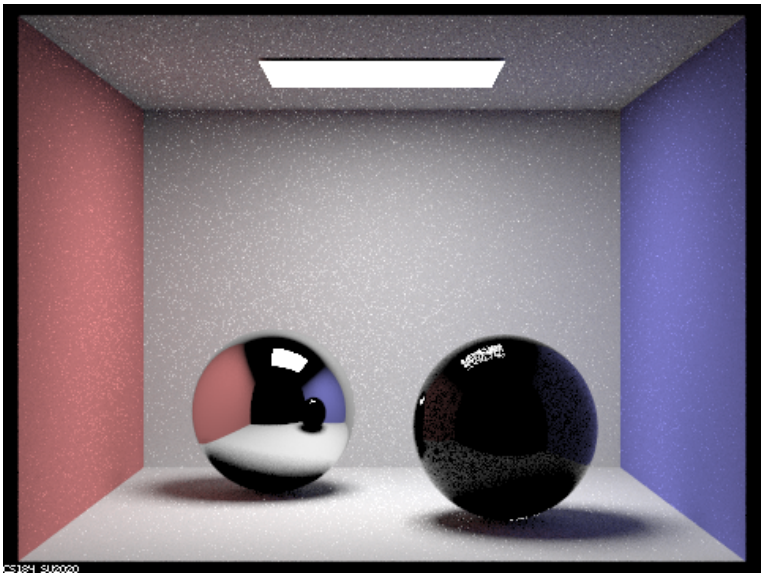


Figure 6: max ray depth = 2

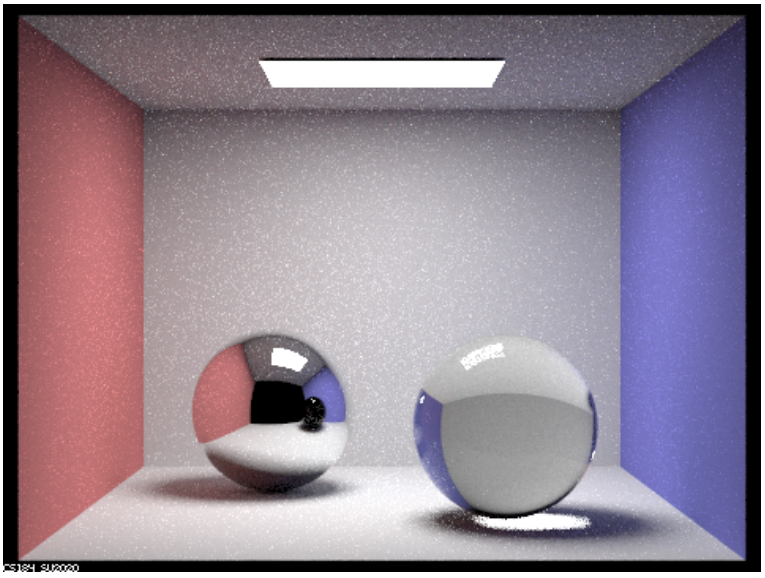


Figure 7: max ray depth = 3

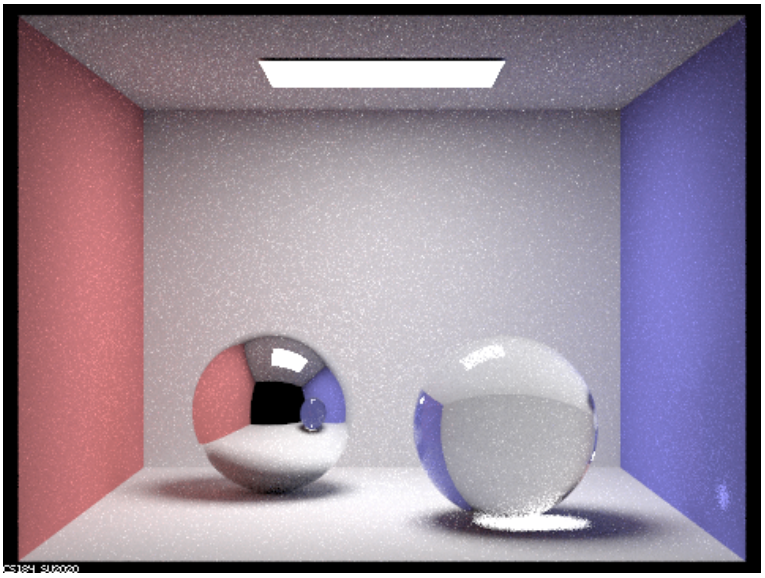


Figure 8: max ray depth = 4

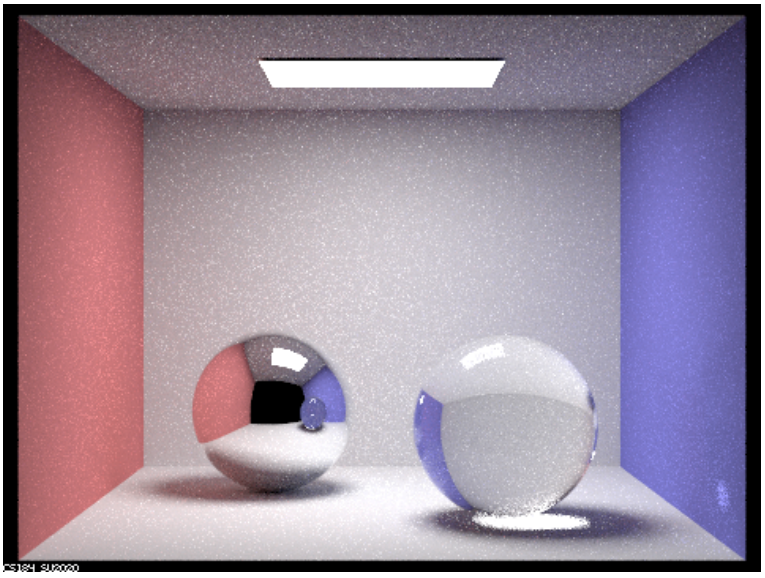


Figure 9: max ray depth = 5

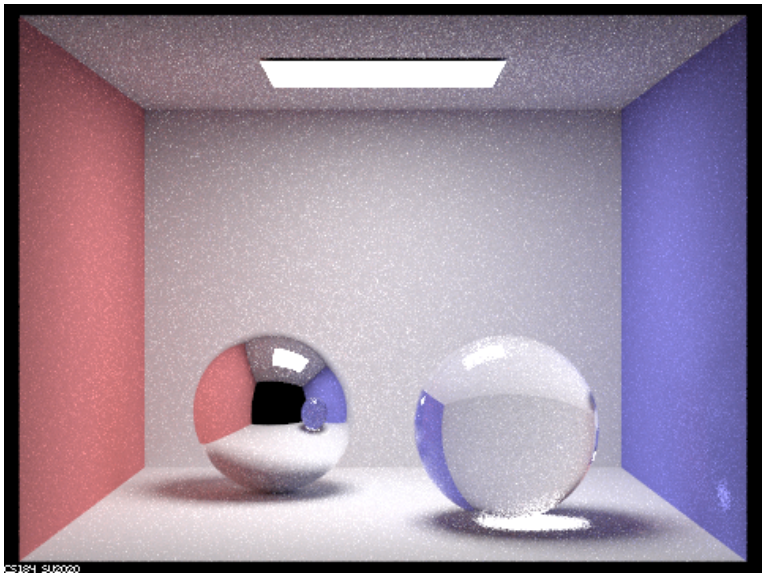


Figure 10: max ray depth = 100

You can see from the figure that: when max ray depth = 1, only the highlight of the sphere can be seen; until max ray depth = 2, the reflected image on the sphere of mirror material can be seen; until max ray depth = 3, the internal refraction image of the sphere of glass material and focused spot in the shadow can be seen; and until max ray depth = 4, the reflection of glass sphere can be seen on the mirror sphere and a light focused plot can be seen on the right hand side wall. And the deeper the max ray depth, the more transparent the glass sphere is.

This is because the color of the mirror material is obtained by reflecting the surrounding scene. To sample the color of the mirror material, the camera ray needs to be reflected at least twice, once by the mirror material and once by the surrounding material, and reach the light source. The color of the glass material is obtained by refracting the surrounding scene. To sample the color of the glass material, the camera ray needs to enter and exit the glass material at with two times of refraction, and then reflect once through the surrounding material to reach the light source. And to sample the focused spot in the shadow can be sampled by reflection once on the floor, refraction twice enter and exit the glass material, and reach the light source while the glass sphere can concentrate the light. When max ray depth = 4, if camera ray reflect once on the mirror material, refract twice to enter and exit the glass material, reflect once on the surrounding scene, and reach the light source, then the reflection of glass sphere can be seen on the mirror sphere. And if camera ray reflect once on the surrounding scene, refract once to enter the material, total reflect once within the and exit the glass material, refract once to exit the glass material, and reach the light source, then an area on the wall get more light exiting from the glass material and form a light focused spot.

# 3 Environment Light

**Idea of environment light:** Among the physically-based rendering methods, if you want to obtain a very good rendering effect, you need to analyze the propagation of light to calculate the global illumination (direct illumination and indirect illumination), but the computational cost required by this method is huge, "Often" cannot support real-time needs. On the contrary, if only the direct lighting of the light source is considered, although the rendering speed is improved, the rendering quality is not satisfactory, and image-based lighting (IBL) is such a method between the two "extremes". The color information on the skybox environment map is regarded as the environment light source, and the sample map is generated by the approximate calculation of the reflection equation, and the final calculation When lighting energy, you only need to sample this texture and then calculate it, which reduces the calculation cost compared to the complete global illumination, and also brings a good improvement in rendering quality.

**Idea of uniform sampling:** Firstly, generate a random direction on the sphere using $sampler\_uniform\_sphere.get\_sample()$ and assign it to in-direction $wi$. Secondly, assign $INF\_D$ to $distToLight$, and assign $1/(\pi)$ to $pdf$. Thirdly, transform $wi$ to xy coordinate on the environment light map using $dir\_to\_theta\_phi(...)$ and $theta\_phi\_to\_xy(...)$. Fourthly, get the radiance of environment light using bilinear interpolation method through $bilerp(...)$.

**Idea of importance sampling:** First of all, compute a pdf map for each pixel of the environment light map using following formula and stored in $pdf\_envmap$:

$$p(x, y) = \frac{E[\lfloor x \rfloor, \lfloor y \rfloor] \sin(\pi \lfloor y \rfloor / h)}{\sum_{i'} \sum_{j'} E[i', j'] \sin(\pi j' / h)}.$$

Figure 11: pdf map formula

Then compute cumulative marginal distribution using following formula and stored in $marginal\_y$:

$$F(j) = \sum_{j'=0}^{j} \sum_{i=0}^{w-1} p(i, j')$$

Figure 12: cumulative marginal distribution formula

And compute cumulative conditional distributions using following formula and stored in $conds\_y$:

$$F(i|j) = P(x < i | y = j) = \int_0^i p(x|y = j)dx = \int_0^i \frac{p(x,j)}{p(j)} dx = \sum_{i'=0}^{i-1} \frac{p(i',j)}{p(j)}.$$

Figure 13: cumulative conditional distribution formula

After that, do the following steps. Firstly, generate a random Vector2D $ab$ using $sampler\_uniform2d.get\_sample()$. Secondly, compute $xy.y$ using $std :: upper\_bound$, $marginal\_y$, and compute $xy.x$ using $conds\_y$, $std :: upper\_bound$, $xy.y$. Thirdly, transform the xy coordinate to direction using $xy\_to\_theta\_phi(...)$, $theta\_phi\_to\_dir(...)$ and assign it to $wi$. Fourthly, assign $INF\_D$ to $distToLight$, and assign $pdf$ using following formula:

$$pdf\_envmap[xy.x + xy.y * w] * w * h/(2 * \pi^2 * sin(\theta))$$

in need of transform the map domain from [0, w]X[0, h] (x, y) coordinate to [0, $\pi$]X[0, 2$\pi$] ($\theta$, $\phi$) coordinate, and to $\omega$ solid angle.

**probability debug.png of field.exr:** The debug images are as following:

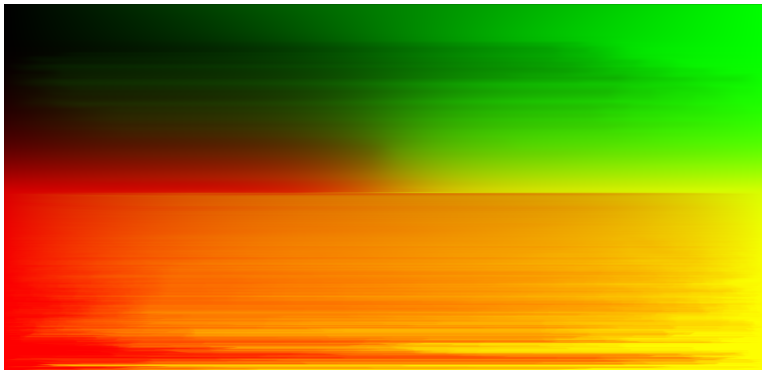


Figure 14: probability debug.png

And the field.exr looked like this:

Figure 15: field.exr

**uniform sampling vs. importance sampling:** The rendered images are as following (field.exr with 4 samples per pixel and 64 samples per light):

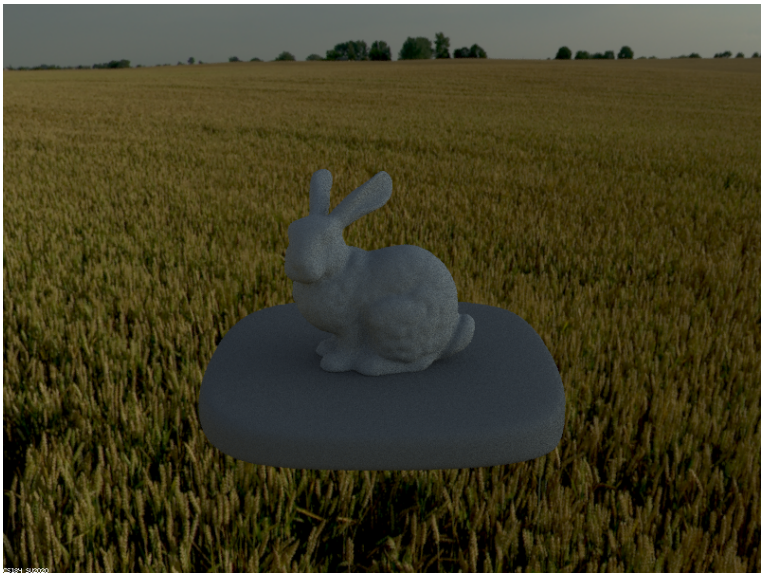


Figure 16: bunny with uniform sampling

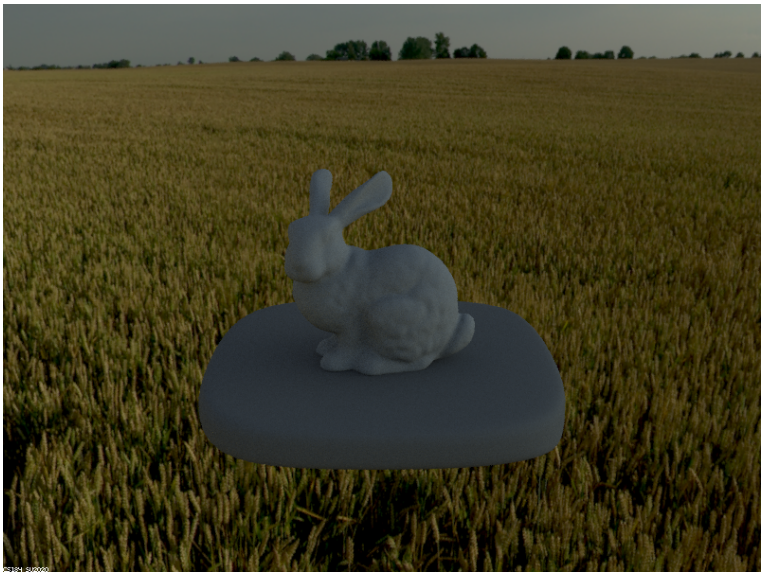


Figure 17: bunny with importance sampling

It can be seen that the noise level of uniform sampling is slightly higher than importance sampling.

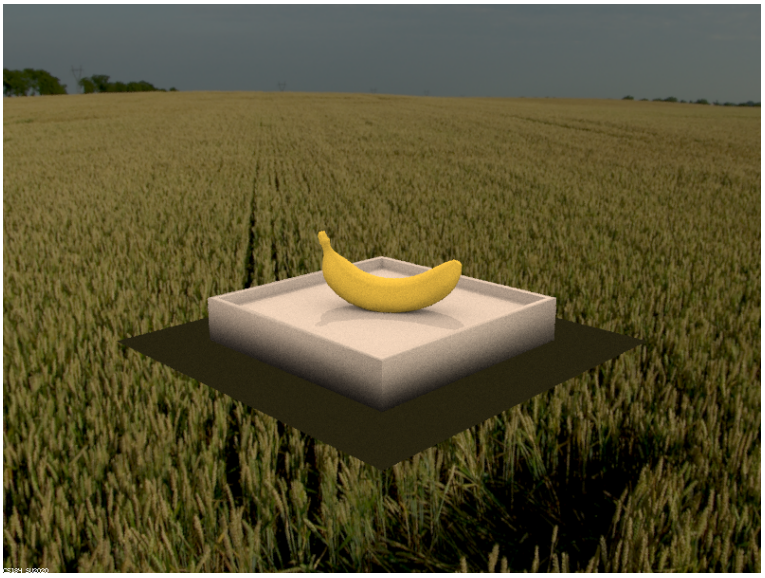


Figure 18: banana with uniform sampling

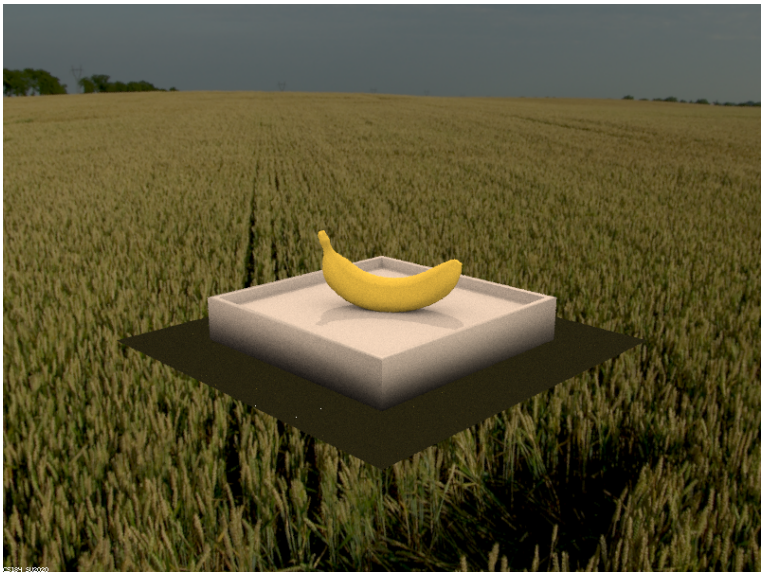


Figure 19: banana with importance sampling

It can be seen that using this dae file, the noise level of both sampling method is similar, both converged.

# 4 Depth of Field

**pinhole camera model vs. thin-lens camera model:** With the pinhole camera model, everything is in focus, while with thin-lens camera model, only objects within a specific distance range is in in focus, other objects are blurred. And with thin-lens camera model, we need to consider lens radius and focal distance.

**Idea of implementation:** Firstly, calculate local camera ray direction as Assignment 6. Secondly, uniformly sample a point $pLens$ on the len disk. Thirdly, calculate the intersection point $pFocus$ between the camera ray and the plane of focus. Fourthly, genrate the ray that originates at $pLens$, with direction towards $pFocus$. Finally, convert the generated ray from camera space to world space and set the $nCLip$ and $fClip$ and return it.

**4 pictures with visibly different depths:** The rendered images are as following (lens radius = 0.05):

Figure 20: focal distance = 1.0

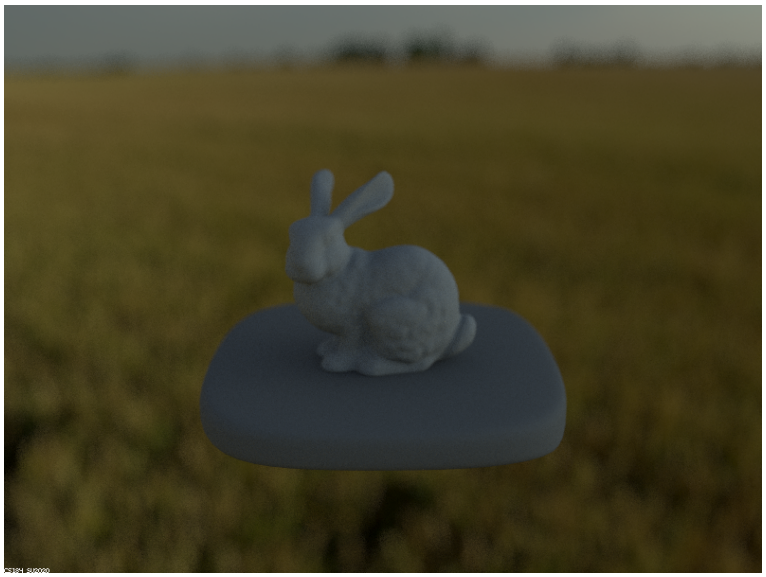


Figure 21: focal distance = 3.0

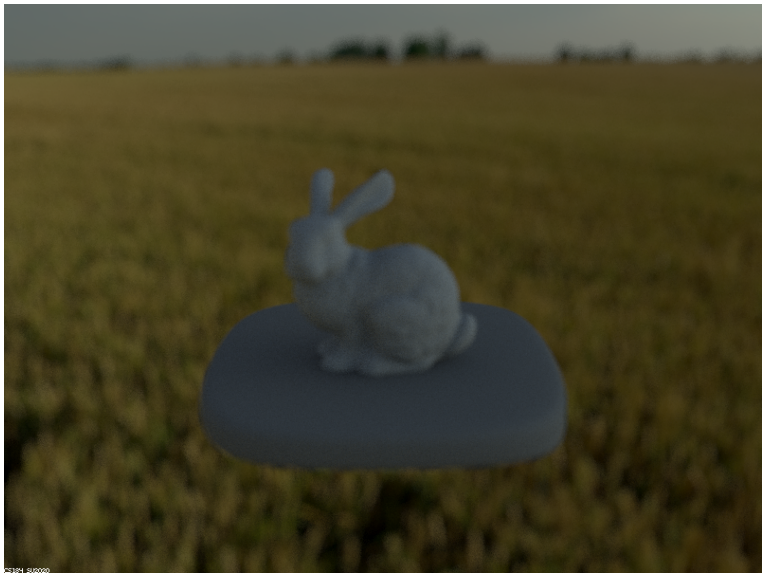


Figure 22: focal distance = 5.0

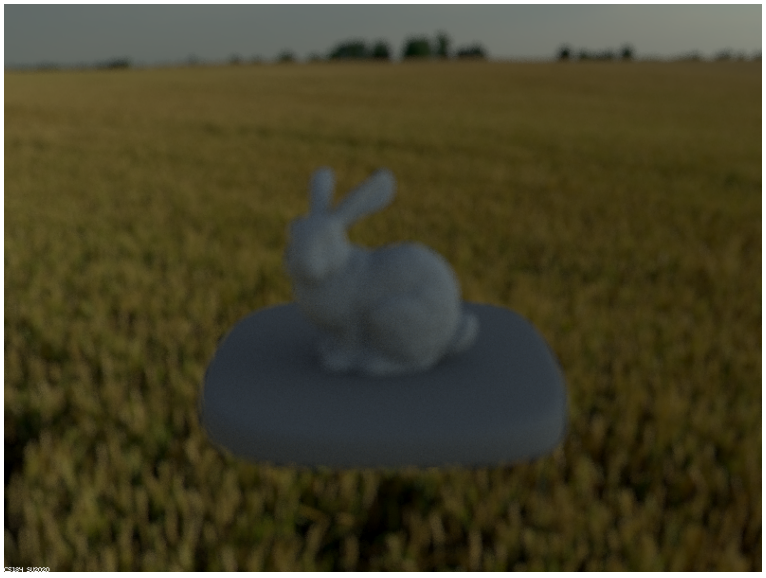


Figure 23: focal distance = 7.0

**4 pictures with visibly different aperture sizes:** The rendered images are as following (focal distance = 4.0)

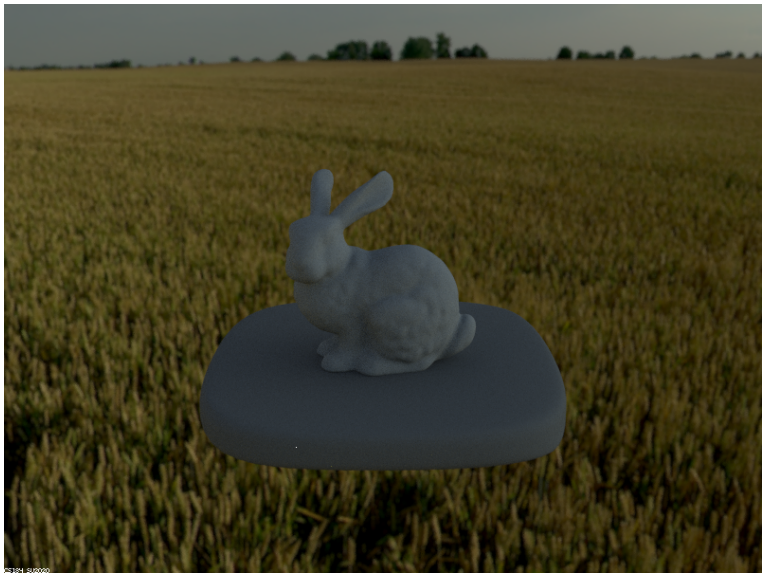


Figure 24: lens radius = 0.01

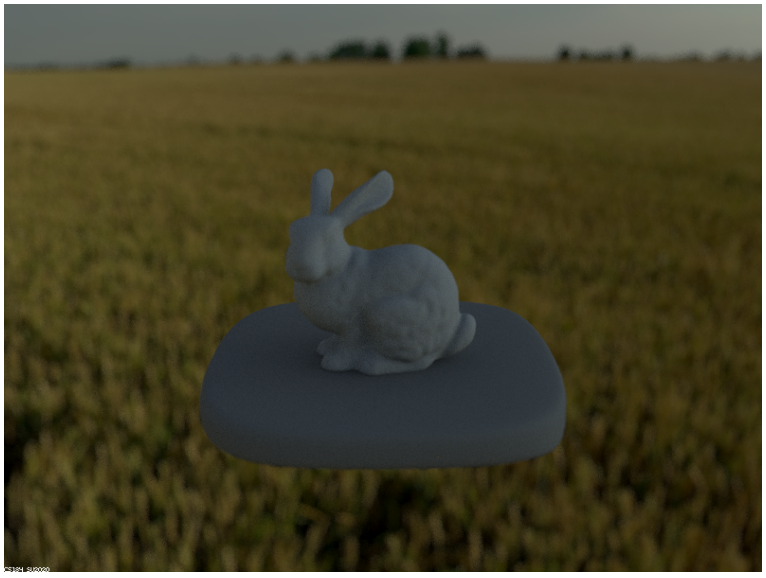


Figure 25: lens radius = 0.03

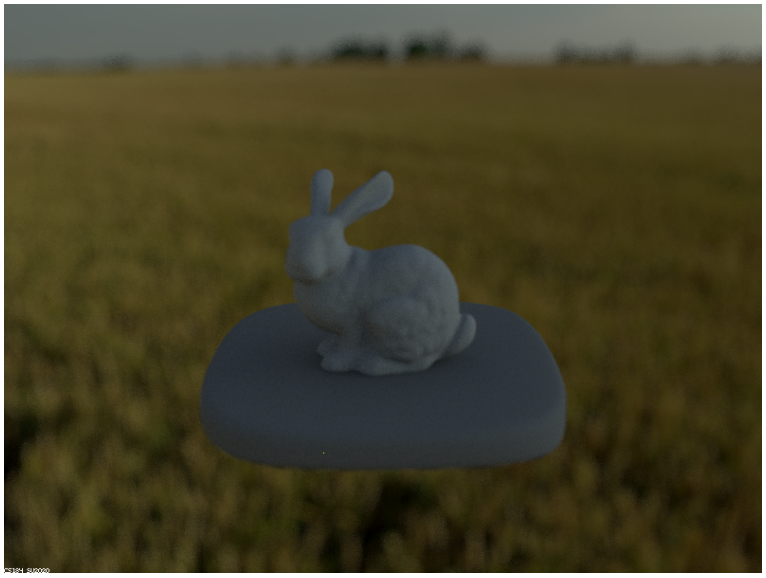


Figure 26: lens radius = 0.05

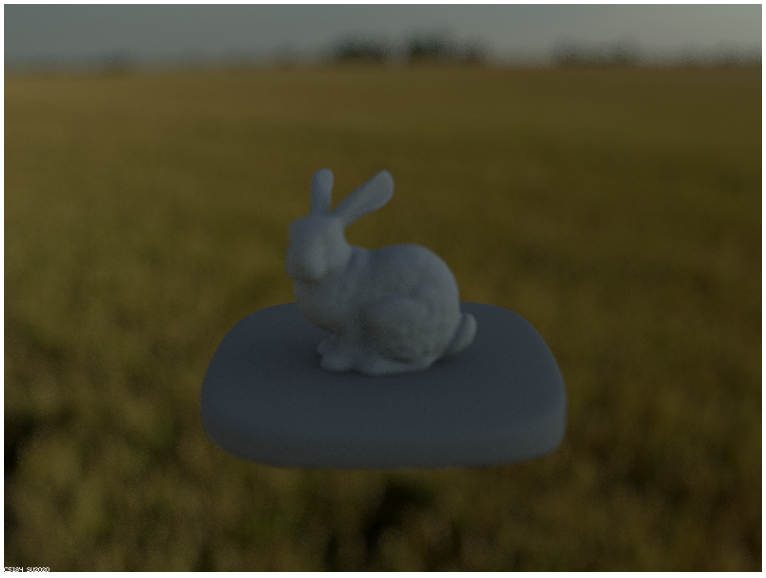


Figure 27: lens radius = 0.07