Assignment 3-2: PathTracer 2

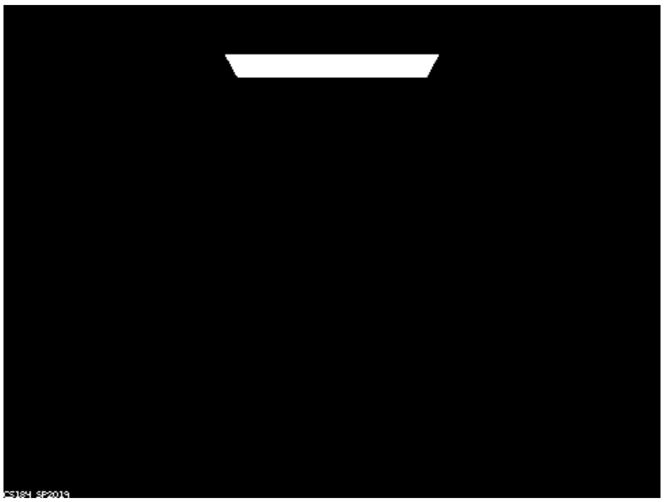
George Zhang

In this project, I rendered mirror, glass, and microfacet materials, explored environment lighting, and implemented a thin-lens camera model to demonstrate depth of field.

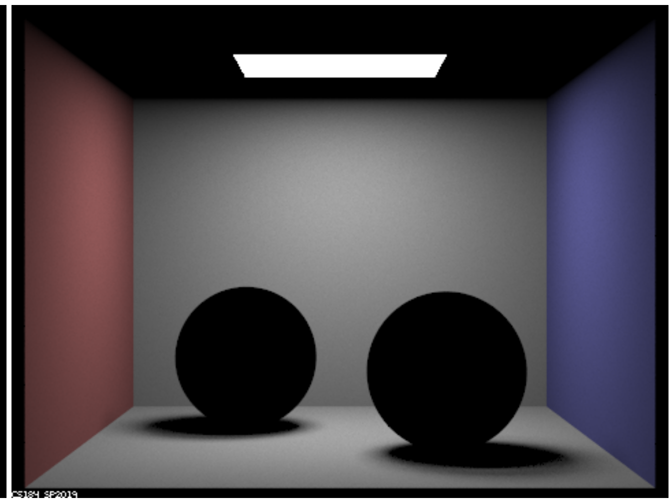
Machine used: Thinkpad T470p.

Part 1: Mirror and Glass Materials

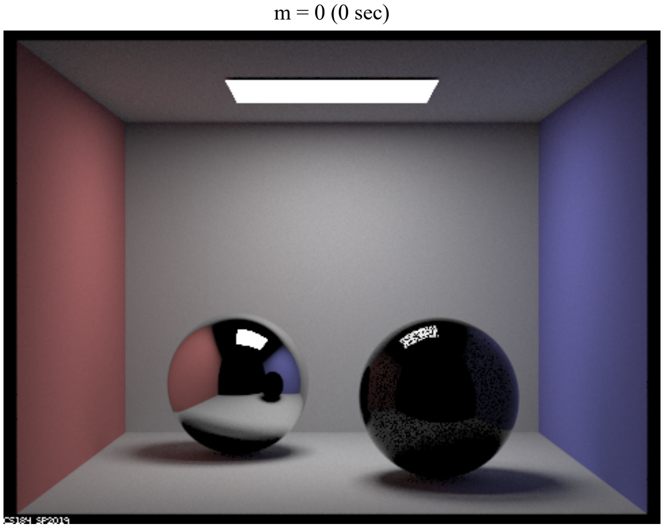
Settings used: -t 8 -s 256 -l 4 -m XXX -r 480 360 (except the last image, which used -s 1024).



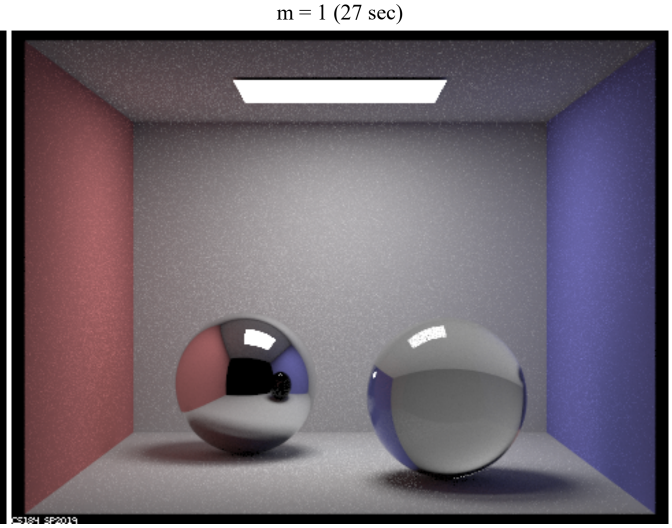
m = 0 (0 sec)



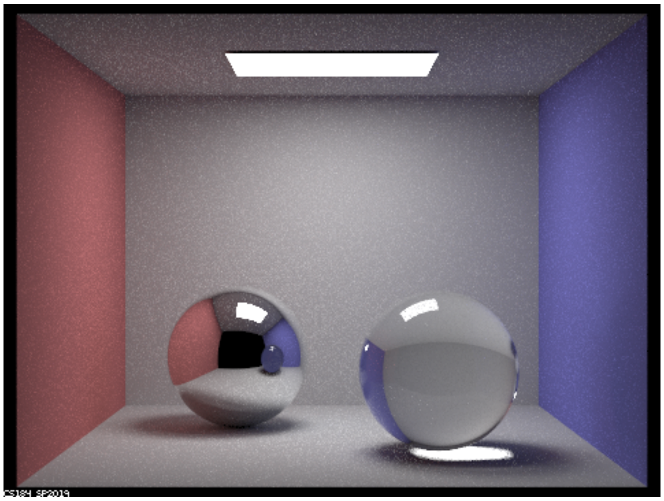
m = 1 (27 sec)



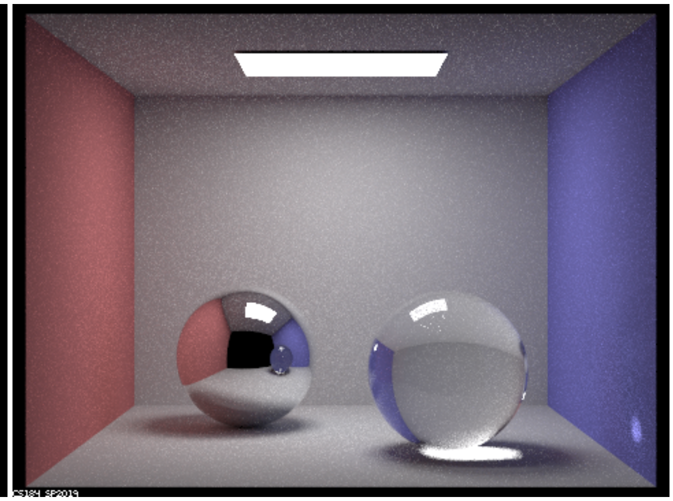
m = 2 (67 sec)



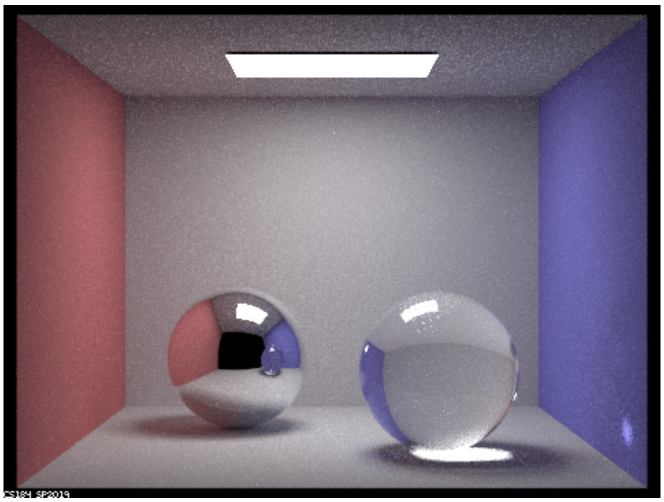
m = 3 (99 sec)



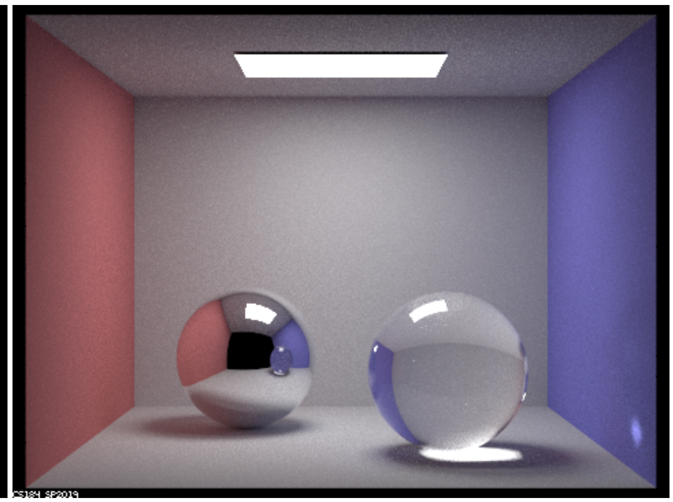
m = 4 (129 sec)



m = 5 (147 sec)



m = 100 (245 sec)



s = 1024, m = 100 (758 sec)

m = 0: Only the light source is seen by the camera.

m = 1: The camera only sees one light bounce meaning that only diffuse (walls) are rendered.

m = 2: The camera now sees two light bounces meaning that light can reflect off of reflective surfaces and hit the light source, thus reflective surfaces can be seen by the camera. Glass objects will not be rendered clearly due to insufficient bounces being unable to capture transmission/refraction. Of course, diffuse is rendered fine, since we have more than one bounce.

m = 3: The camera now sees three light bounces meaning that light can now hit glass objects, refract twice, and hit the light source, thus the camera can now capture clear glass images.

m = 4: The camera can now capture a circle of light under the glass sphere. This is due to one reflection off the floor, two refractions in the glass sphere, and one hit on the light source.

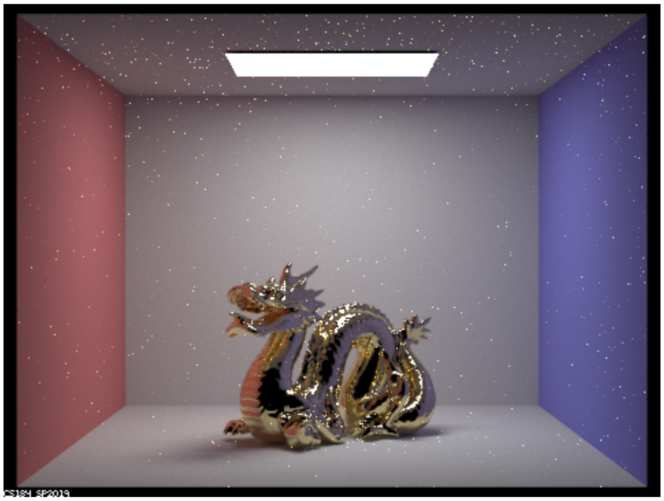
m = 5: The camera can now capture a circle of light on the right blue wall as well as more light near the bottom of the glass sphere. This is due to one reflection off the right blue wall, three refractions in the glass sphere, and one hit on the light source.

m = 100: The max ray depth is increased so much that Russian Roulette will likely terminate the bouncing before the max ray depth is reached.

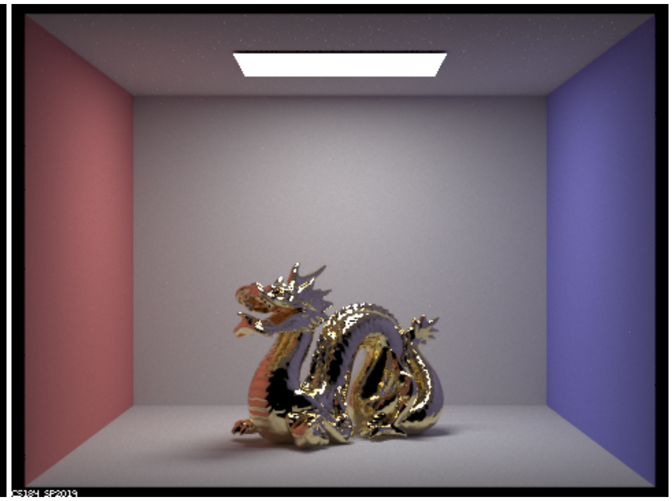
Part 2: Microfacet Material

For Part 2, in general, I used a Visual Studio command of the form: -t 8 -s 1024 -l 4 -m 100 -r 480 360 $(ProjectDir)..\dae\sky\CBdragon_microfacet_au.dae

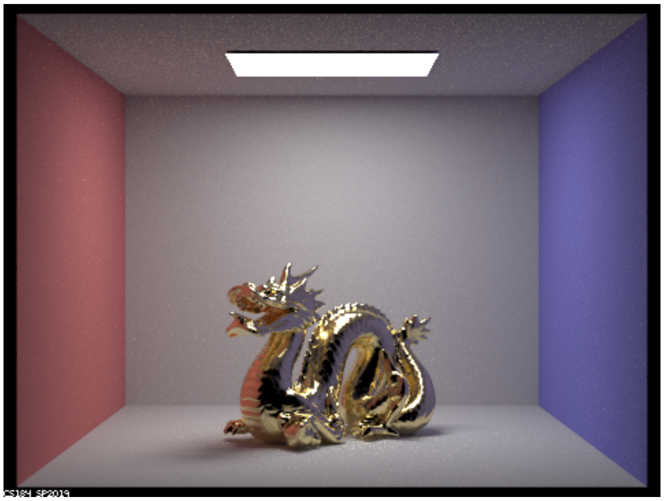
Settings used: -t 8 -s 1024 -l 4 -m 100 -r 480 360 unless stated otherwise.



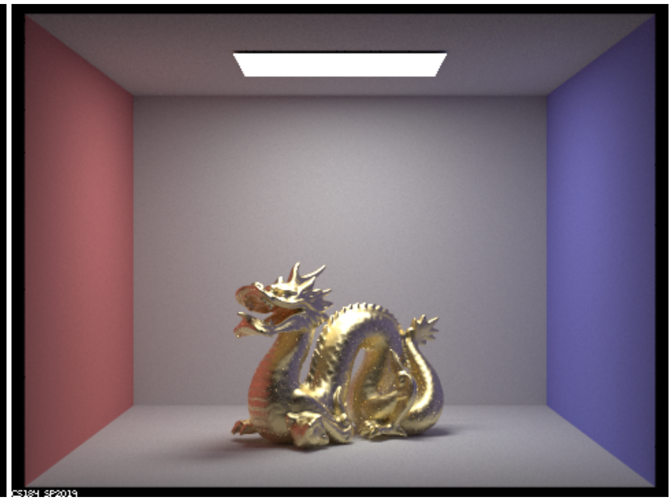
α = 0.005 (602 sec)



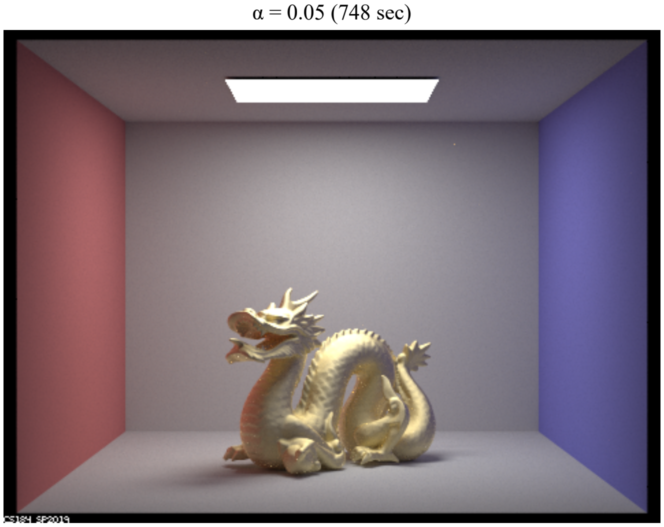
α = 0.005, -s 65536 samples/pixel



α = 0.05 (748 sec)



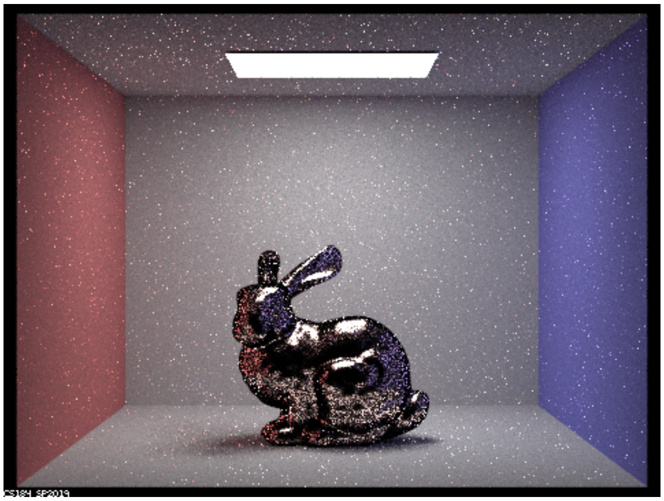
α = 0.25 (738 sec)



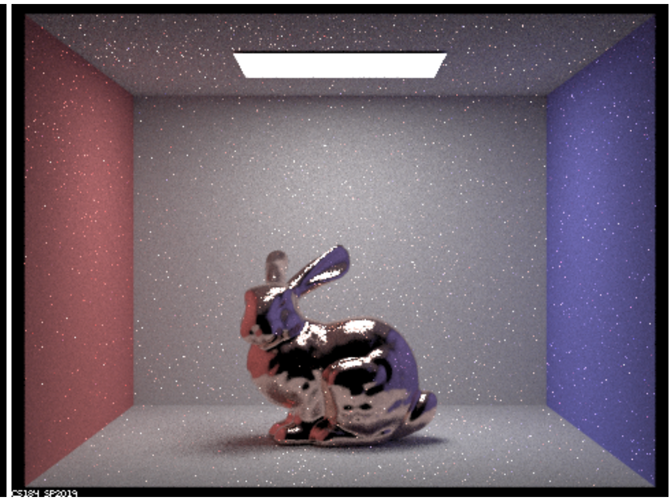
α = 0.5 (604 sec)

As α of the object decreases, not only does the object appear shinier (smoother macro surface), but also more white specks appear. These white specs can be eliminated by increasing the -s parameter (samples/pixel). For α = 0.005, I increased the samples/pixel to 65536 to eliminate most of the white specks. The prominence of white specks is correlated with decreasing α, which suggests that decreasing α causes white specks. This may be due to the object being shinier (more reflective to light).

Settings used: -t 8 -s 64 -l 1 -m 5 -r 480 360.



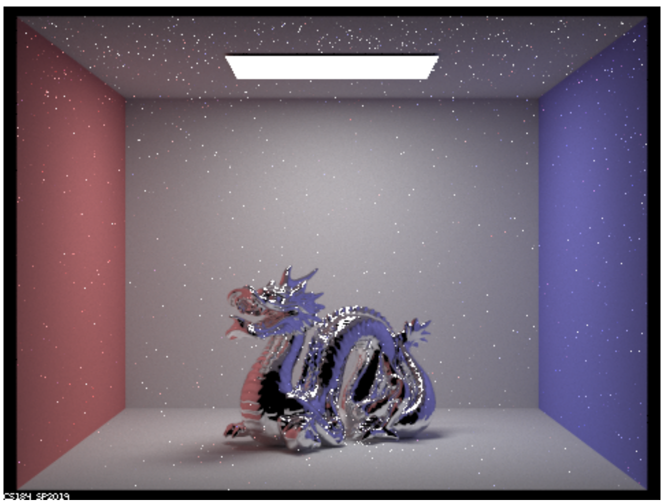
Cosine hemisphere sampling (22 sec)



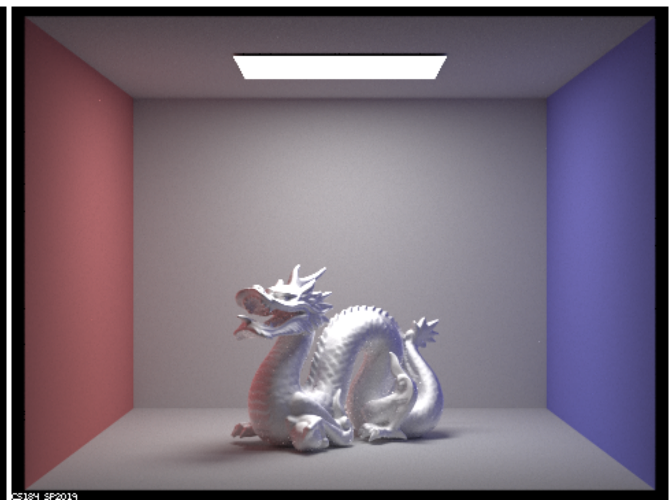
Importance sampling (23 sec)

Cosine hemisphere sampling produces a noisier image when compared to importance sampling. This causes parts of the bunny that would be otherwise bright when rendered using importance sampling to be dark when rendered using cosine hemisphere sampling. As mentioned in the project spec, the cosine hemisphere sampling provided for us is suitable for importance sampling diffuse BRDFs. However, this is not the case for microfacet BSDFs whose normal distribution function (NDF) can be defined using a Beckmann distribution.

Settings used: -t 8 -s 1024 -l 4 -m 100 -r 480 360.



$\alpha = 0.005$ (589 sec)



$\alpha = 0.5$ (688 sec)

Metal chosen: silver

According to https://refractiveindex.info/, for silver, the η and k values at 0.614 μm (red), 0.549 μm (green), and 0.466 μm (blue) are:

$\eta = (0.059193, 0.059881, 0.047366)$

$k = (4.1283, 3.5892, 2.8132)$

Part 3: Environment Light

For Part 3, in general, I used a Visual Studio command of the form: -t 8 -s 4 -l 64 -e $(ProjectDir)../exr/field.exr $(ProjectDir)..\dae\sky\bunny_microfacet_cu_unlit.dae

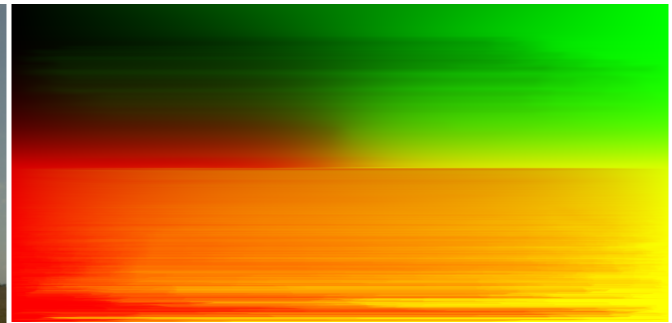
In uniform sampling, a random direction (wi) is sampled from the sphere surrounding the hit point. Bilinear interpolation is then used to compute the radiance on this point from the sampled direction.

In importance sampling, sampling of wi is biased towards light sources (from which the radiance is comparatively large), thus hit areas with strong lighting will be less noisy, simple more wi samples come from the vicinity of light sources.

I modified CBempty.dae poorly by slightly decreasing count in ceiling-mesh-positions-array. This allowed me to see field.exr unblocked.

To normalize pdf_envmap, simply divide each value in pdf_envmap by sum. marginal_y stores the cumulative marginal distribution for y. Thus, we sum all values of pdf_envmap with the same y (varying x). I chose to also keep track of the (non-cumulative) marginal distribution for y in marginal_y0. marginal_y0 made calculating values for conds_y easy, since I could just divide by values in marginal_y0.

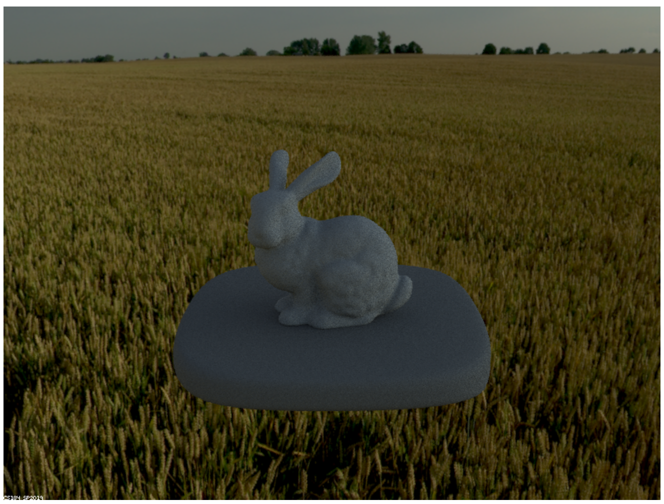
field.exr for modified CBempty.dae



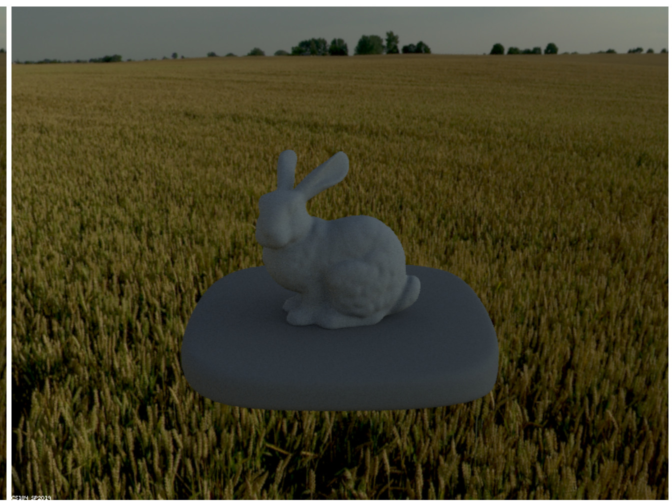
Probability debug for field.exr

This matches the project spec.

Settings used: -t 8 -s 4 -l 64.



Uniform sampling: CBbunny_unlit (17 sec)



Importance sampling: CBbunny_unlit (16 sec)

Looking closely at both images, I see that the image generated by uniform sampling is slightly more noisy than that generated by importance sampling (black dots more prominent in uniformly sampled image).

Uniform sampling: CBbunny_microfacet_cu_unlit (18 sec)

Importance sampling: CBbunny_microfacet_cu_unlit (17 sec)

Looking closely at both images, I see that the image generated by uniform sampling is slightly more noisy than that generated by importance sampling (black dots more prominent in uniformly sampled image, especially around the cheek, ear, and hind of the bunny).

Part 4: Depth of Field

For Part 4, in general, I used a Visual Studio command of the form: -t 8 -s 256 -l 4 -m 8 -b 0.125 -d 1.8 -c $(ProjectDir)..\dae\sky\CBdragon_cam_settings.txt $(ProjectDir)..\dae\sky\CBdragon.dae

Pinhole camera model vs. thin-lens camera model: The pinhole camera model has linear perspective, but does not involve focusing. The thin-lens camera model involves the Thin Lens equation, magnification, and depth of field. Moreover, a nonideal lens may not allow rays to converge perfectly. We can think of a pinhole camera as a thin-lens camera with zero aperture.

Settings used: -t 8 -s 256 -l 4 -m 8.

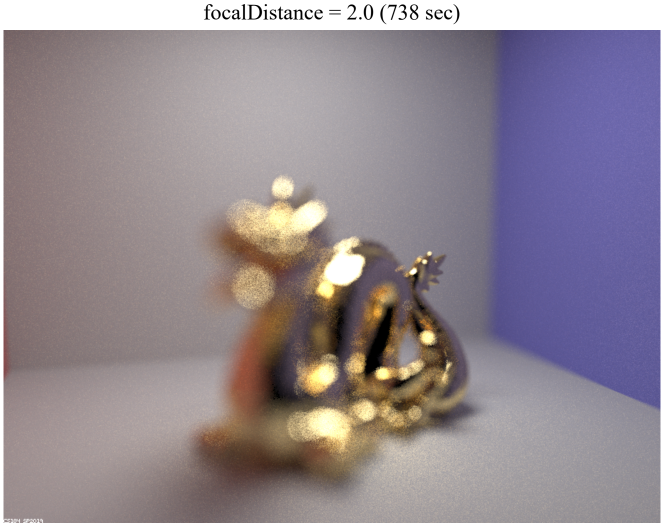
lensRadius fixed at 0.125.


focalDistance = 1.6 (747 sec)


focalDistance = 1.8 (802 sec)


focalDistance = 2.0 (738 sec)

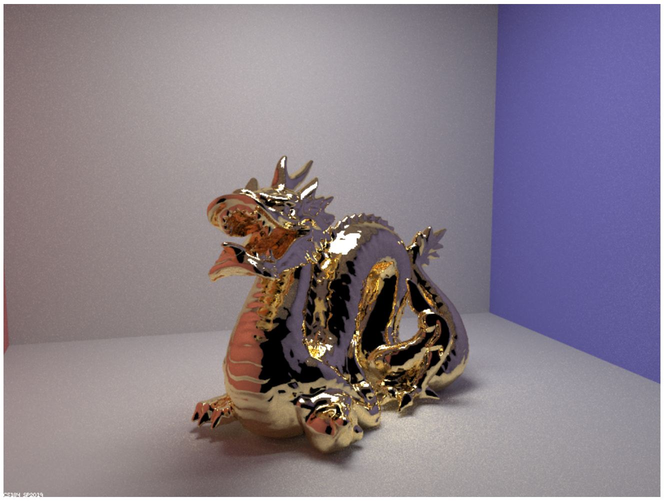

focalDistance = 2.2 (734 sec)

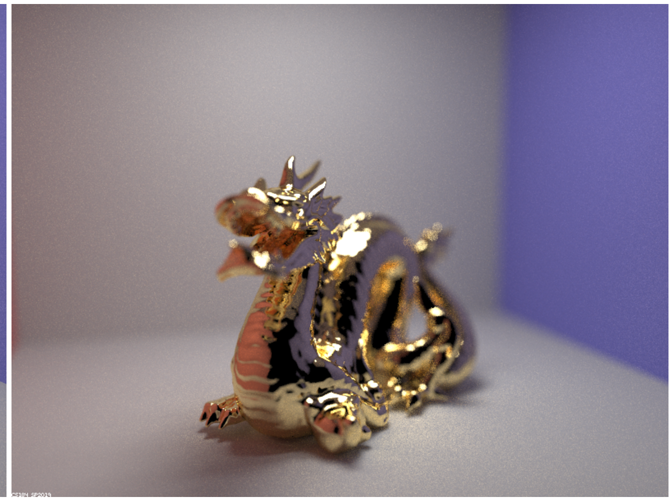

focalDistance = 2.4 (720 sec)

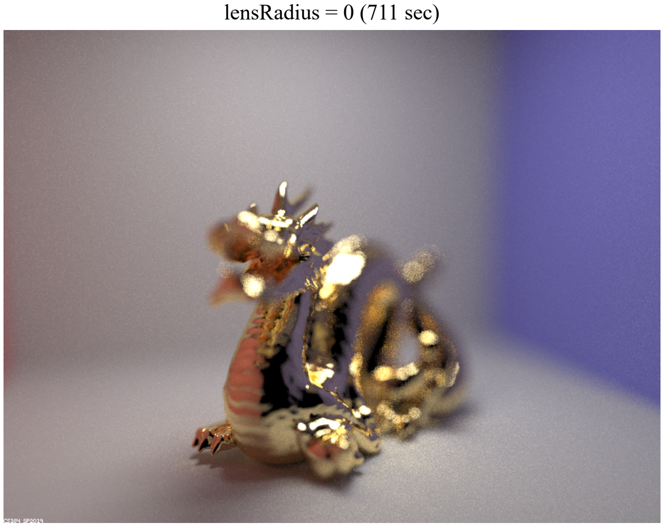
As expected, with a fixed lensRadius of 0.125, as focalDistance is increased, the image appears focused at a further distance.

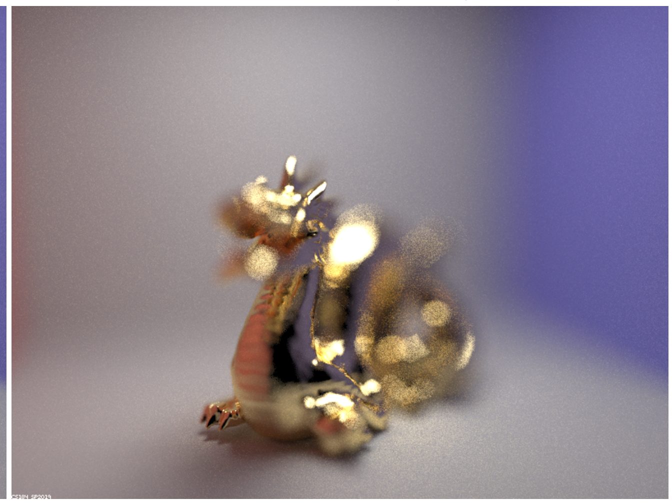
Settings used: -t 8 -s 256 -l 4 -m 8.

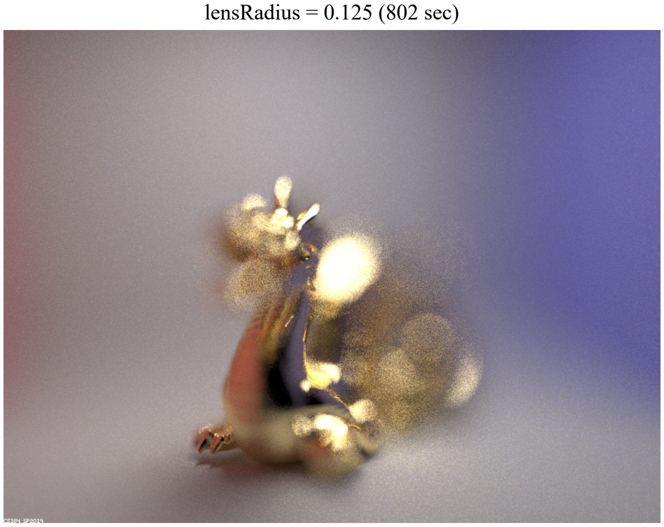
focalDistance fixed at 1.8.

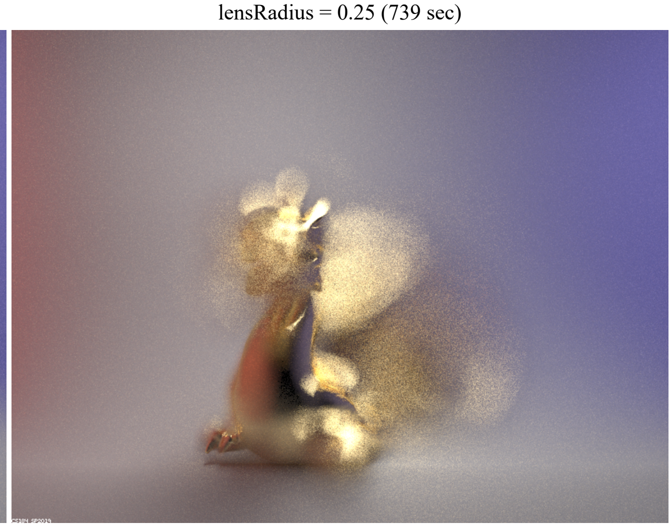

lensRadius = 0 (711 sec)


lensRadius = 0.0625 (725 sec)


lensRadius = 0.125 (802 sec)


lensRadius = 0.25 (739 sec)


lensRadius = 0.5 (753 sec)


lensRadius = 1 (777 sec)

As expected, with a fixed focalDistance of 1.8, as lensRadius is increased, the image appears blurrier, especially at further distances.