CS 184: Computer Graphics and Imaging, Spring 2023

Project 3-2:

Brian Santoso, CS184 Team: #1-lana-del-rey-fan-2

Webpage: https://briansantoso.github.io/project-webpages-sp23-BrianSantoso/proj3-2/index.html

In this project I implemented reflective/refractive BSDF's (Part 1) and depth of field with a thin-lens camera model (Part 4).

Part 1

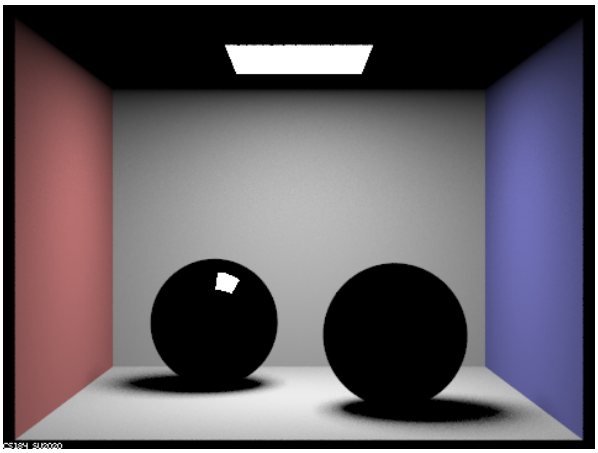
In this project I implemented reflective/refractive BSDF's. I implemented a helper reflection function to compute ray-normal reflections. For the mirror material, the bsdf for any sample has a pdf of 1 since a perfect mirror will just sample the perfect reflection across the normal of the surface of intersection. The bsdf returns reflectance *divided by abs_cos_theta(*wi)* in order to negate the effect of the lambertian falloff term that at_least_one_bounce_radiance includes in the product.

For the glass material, I similarly implement a refraction helper function that computes the refraction of a ray given the index of refraction and returns true iff there is no complete internal reflection. If there is total internal reflection, the bsdf will always mimic, the reflective bsdf. Otherwise, I compute the Schlick's reflection coefficient and compute the reflection or refraction of wo with probability p and 1-p since a glass material results in both a reflection and refraction and we can only return one of them. The reflection of wo in the glass material scales the pdf and bsdf of the mirror material by the refractive coeffiecient. The refraction wo has a pdf of the complement of the refractive coefficient and returns (1-R) * transmittance / abs_cos_theta(*wi) / eta^2 where eta^2 is the same as in the refraction function.

- **Show a sequence of six images of scene `CBspheres.dae` rendered with `max_ray_depth` set to 0, 1, 2, 3, 4, 5, and 100. The other settings should be at least 64 samples per pixel and 4 samples per light. Make sure to include all screenshots.**

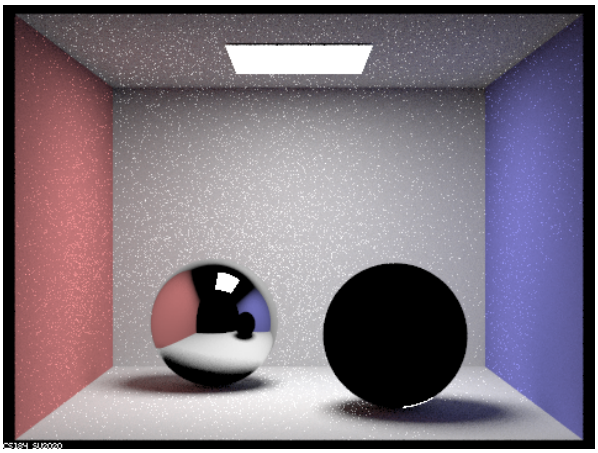
Max Ray Depth 0

`./pathtracer -t 8 -s 64 -l 4 -m 0 -r 480 360 -f p1_0.png ../dae/sky/CBspheres.dae`



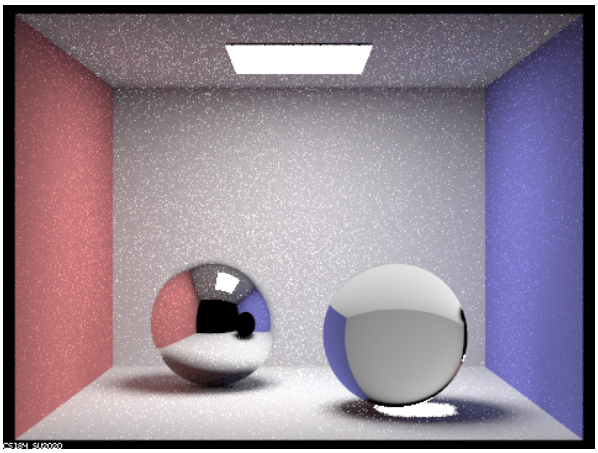
Max Ray Depth 1

`./pathtracer -t 8 -s 64 -l 4 -m 1 -r 480 360 -f p1_1.png ../dae/sky/CBspheres.dae`



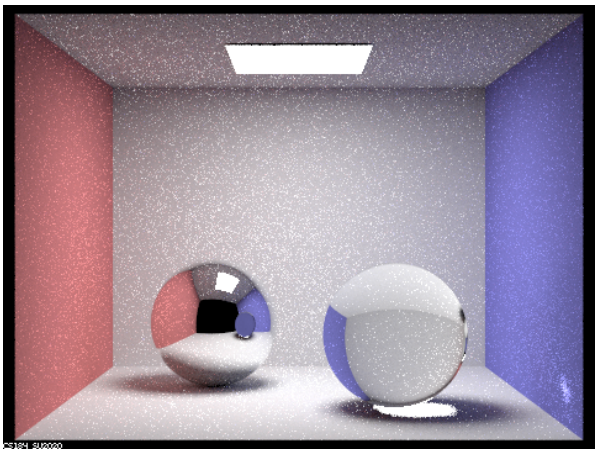
Max Ray Depth 2

`./pathtracer -t 8 -s 64 -l 4 -m 2 -r 480 360 -f p1_2.png ../dae/sky/CBspheres.dae`



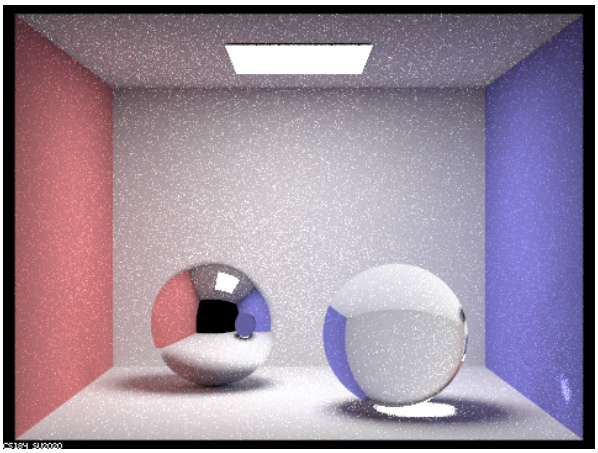
Max Ray Depth 3

`./pathtracer -t 8 -s 64 -l 4 -m 3 -r 480 360 -f p1_3.png ../dae/sky/CBspheres.dae`



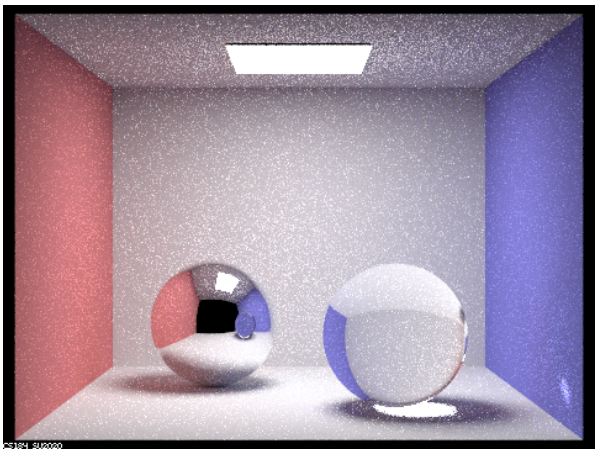
Max Ray Depth 4

`./pathtracer -t 8 -s 64 -l 4 -m 4 -r 480 360 -f p1_4.png../dae/sky/CBspheres.dae`



Max Ray Depth 5

`./pathtracer -t 8 -s 64 -l 4 -m 5 -r 480 360 -f p1_5.png../dae/sky/CBspheres.dae`



Max Ray Depth 100

`./pathtracer -t 8 -s 64 -l 4 -m 100 -r 480 360 -f p1_100.png ../dae/sky/CBspheres.dae`

Point out the new multibounce effects that appear in each image. Explain how these bounce numbers relate to the particular effects that appear. Make sure to include all screenshots.

With a max ray depth of 0, we only get the zero-bounce lighting, i.e. the radiance directly from the lights to the camera

With a max ray depth of 1, the spheres appear black. This is because the spheres are reflective and refractive, and require more than 1 bounce for the light to reflect off of and enter (refract) the spheres' surfaces, respectively.

With a max ray depth of 2, we begin to see a reflection in the reflective sphere. The walls are visible in the reflection (albeit with “direct” lighting, since the rays run out of bounces after reflecting off the sphere and hitting the wall), however, inside the reflection the second sphere is still black because it requires more than 3 bounces for the light to reflect off the reflective sphere’s surface and refract through both of the the refractive sphere’s surfaces. The refractive sphere is still black because it requires more than 2 bounces for light to enter and exit the refractive sphere’s body and then bounce into the camera. A very small white highlight appears underneath the refractive sphere because there is a region on the floor under the refractive sphere where light can bounce off one wall, land on that region, and bounce into the camera, instead of passing directly through the refractive sphere.

With a max ray depth of 3, we begin to see “indirect” lighting of the walls inside the reflection of the first sphere. Again, inside the reflection the second sphere is still black because it requires more than 3 bounces for the light to reflect off the reflective sphere’s surface and refract through both of the the refractive sphere’s surfaces. The refractive sphere begins to show minimal refraction, with the refracted view of the room through the sphere showcasing “direct” lighting. This is because there is only enough bounces for the light to enter and exit the refractive sphere and bounce once off of a wall and enter the camera. A white circular region appears under the refractive sphere because it is a focal area where rays from the area light pass through both surfaces of the refractive sphere and are refracted into focal area.

Allowing for 1 more bounce with a max ray depth of 4, we see the newfound effects of the refractive sphere from max ray depth of 3 inside the reflection of the left sphere. The view of the room from the refractive sphere begins to exhibit “indirect” lighting. Furthermore, a small bright region on the blue wall becomes visible—this is due to the light from the newfound focal area underneath the refractive sphere bouncing onto the blue wall.

With a max ray depth of 5, the bright spot on the blue wall becomes visible inside the reflection of the left sphere. We have unlocked all key features of the scene inside the reflection of the left sphere itself. The scene approaches closer to the answer.

With max ray depth of 100, no new key features are unlocked, and the scene approaches closer to the answer.

Part 4

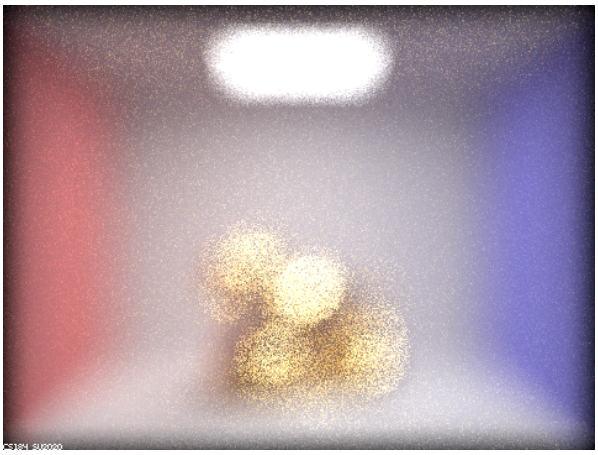
I implement the thin-lens camera model. For a given unnormalized image coordinate (x, y) in [0, 1], generate a sampled thin lens ray. First, I compute the point on the sensor (in camera space) in which a standard pinhole ray would pass through. Then, I intersect this ray with the focal plane to get the focal point. Using the focal point, I leverage the property that all rays from the focal point passing through the lens land on the same point on the sensor—this helps me generate the ray starting from a sampled point on the lens, in the direction of the focal point. I return this sampled thin-lens ray ray after transforming it from camera space into world space.

- **In a few sentences, explain the differences between a pinhole camera model and a thin-lens camera model.**

A pinhole camera models an infinitesimally small point in which light can pass through the “pinhole”. In this model, the radiance of a given point on the sensor corresponds to exactly one light ray that passes through the pinhole from the scene. On the other hand, a thin-lens camera model replaces the infinitesimally small pinhole with a refractive lens with non-zero area. This means that multiple rays which pass through the lens will contribute to the radiance of a given point on the sensor and this manifests into the presence of a focal point and depth of field effect. This is because outgoing rays from a given point on a sensor will converge on a single point in space in front of the lens (the focal point), or diverge into different parts of the scene if no object is present near the focal point.

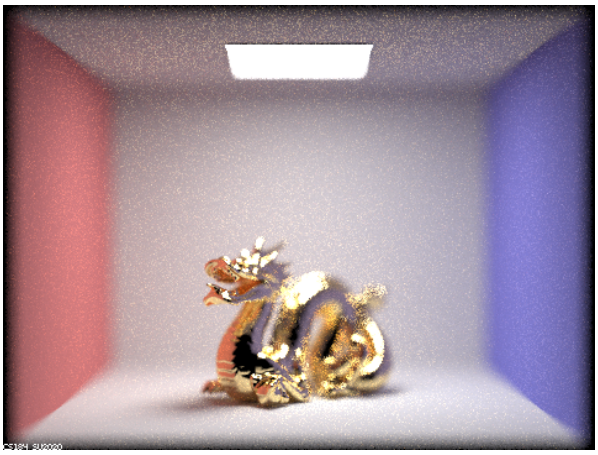
- **Show a "focus stack" where you focus at 4 visibly different depths through a scene. Make sure to include all screenshots.**

With a focal point in front of the dragon, the dragon appears blurry because the rays from a given sensor point pass through the focal point and diverge into difference parts of the scene. With a focal point on the dragon, the dragon appears clear, with a depth of field effect blurring the other parts of the scene. With a focal point behind the dragon, the dragon appears blurry because the rays from a given sensor point intersect the dragon at different points in the scene before reaching the focal point.



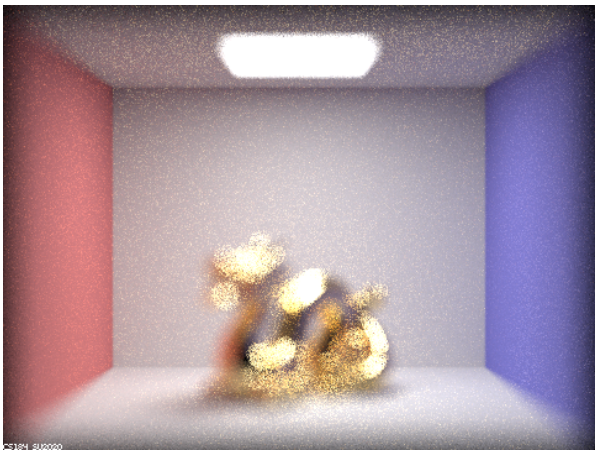
Focal distance of 3

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.23 -d 3 -f focusstack_3.png ../dae/sky/CBdragon.dae
```



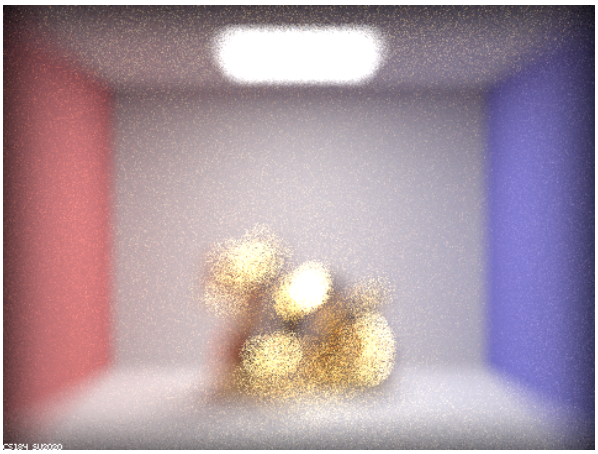
Focal distance of 4.5

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.23 -d 4.5 -f focusstack_4.5.png ../dae/sky/CBdragon.dae
```



Focal distance of 6

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.23 -d 6 -f focusstack_6.png ../dae/sky/CBdragon.dae
```

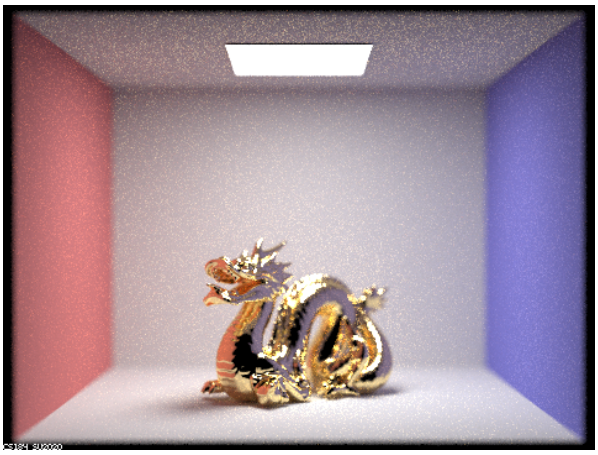


Focal distance of 7.5

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.23 -d 7.5 -f focusstack_7.5.png ../dae/sky/CBdragon.dae
```

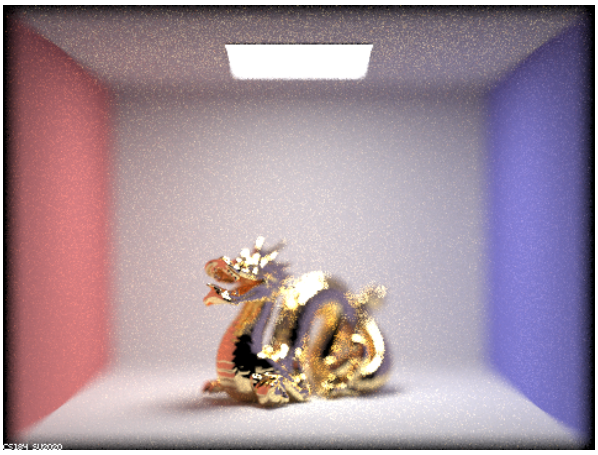
- **Show a sequence of 4 pictures with visibly different aperture sizes, all focused at the same point in a scene. Make sure to include all screenshots.**

With a tiny aperture size of 0.1, the scene appears closer to a pinhole camera render since the rays from a given point on the sensor pass through almost the same point on the lens. With an aperture size of 0.23, the scene is identical to the second render in the previous part, and the result is explained above. With a large aperture size, the rays from a given sensor point have trouble converging on the scene because the lens radius is large, resulting in a large depth of field effect.



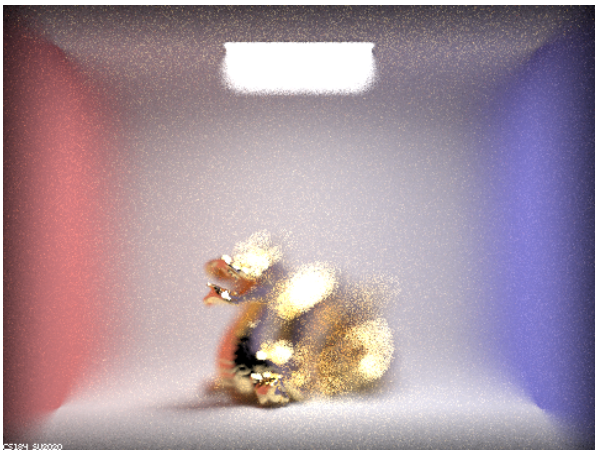
Aperture size of 0.1

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.1 -d 4.5 -f aperture_0.1.png ../dae/sky/CBdragon.dae
```



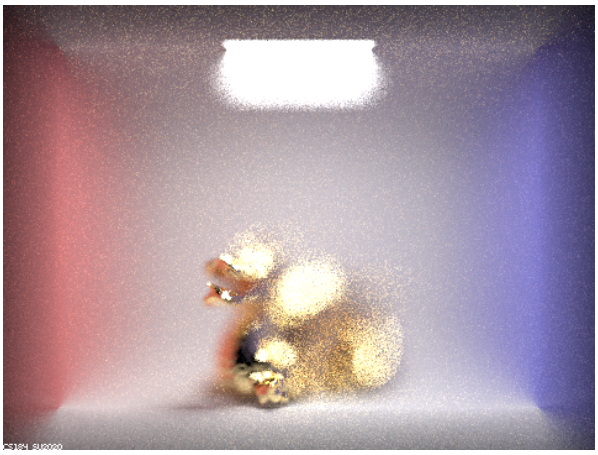
Aperture size of 0.23

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.23 -d 4.5 -f aperture_0.23.png ../dae/sky/CBdragon.dae
```



Aperture size of 0.73

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 0.73 -d 4.5 -f aperture_0.73.png ../dae/sky/CBdragon.dae
```



Aperture size of 1.23

```
./pathtracer -t 8 -s 128 -a 64 0.05 -r 480 360 -m 12 -l 4 -b 1.23 -d 4.5 -f aperture_1.23.png ../dae/sky/CBdragon.dae
```