CS184 Project 3-2

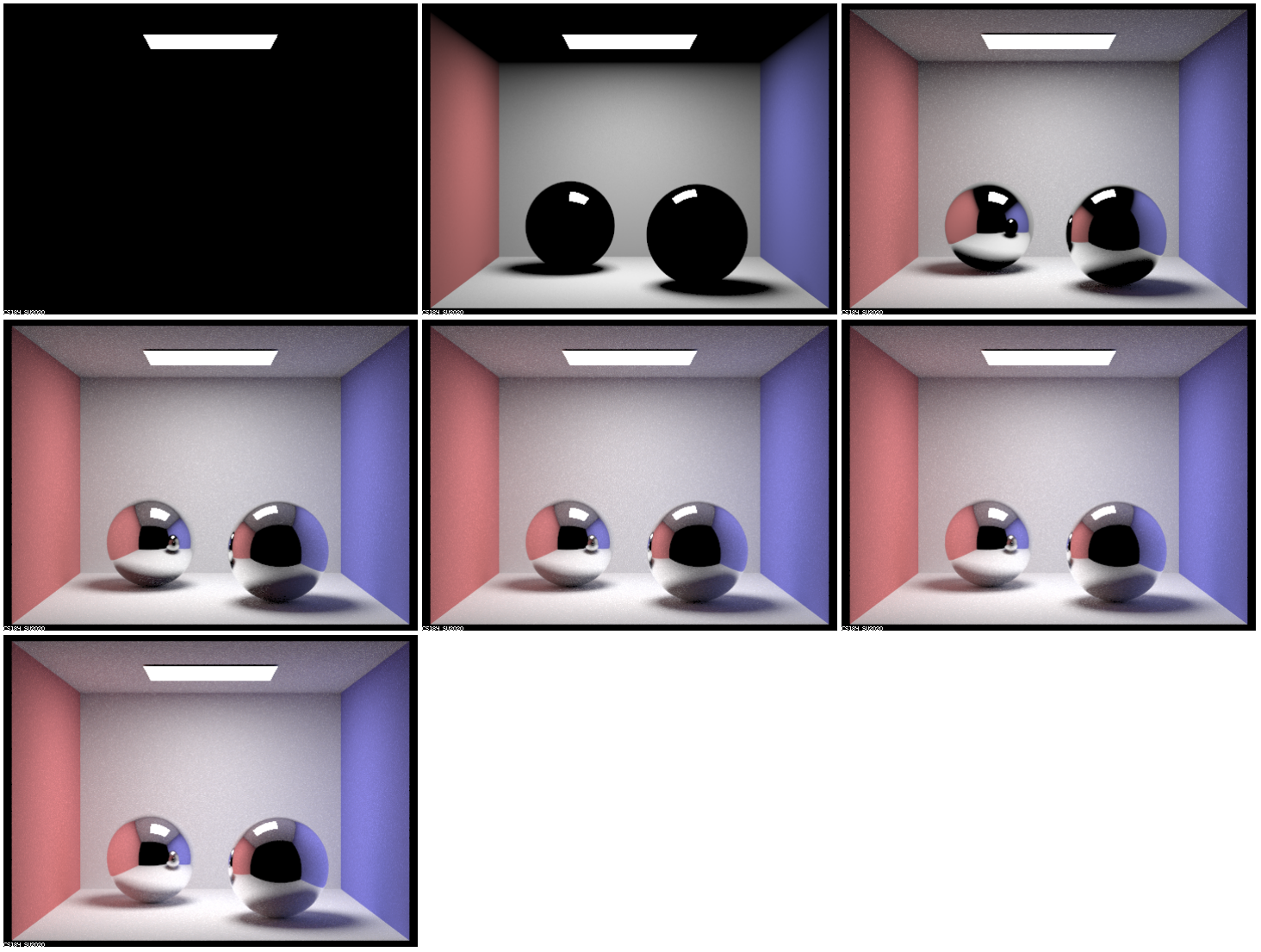
Matthew Tang, Kaitlyn Lee

Chosen Parts: Parts 1 and 4

https://cal-cs184-student.github.io/sp22-project-webpages-kaitlynlee/proj3-2/index.html

Part 1

In Part 1, we implemented code to support reflection and refraction for glassy materials. Below is the renders for depths 0, 1, 2, 3, 4, 5, 100.



At depth 0, we just only see the light source which is to be expected. At depth 1, we see the first bounce which bounces on the walls of the room. The spheres only show black with a bit of white from the light source since there is only a single bounce of reflection/refraction from the light source. At depth 2, the spheres now show reflection/refraction from the walls and surrounding objects since it has the second bounce. The reflection in the sphere contains the other sphere which is dark still until depth 3. The differences between depths 4, 5, and 100 are hard to notice by the naked eye due to their very small contributions, but generally the pictures are slightly brighter due to the extra bounces added.

Implementation

To implement BSDF reflect, we can simply flip the signs on the x and y components. For MirrorBSDF::sample_f we can call reflect to get a wi and return reflectance / abs_cos_theta(*wi).
Refraction can be implemented using the steps described in the spec. We check dot(wo, N) > 0 for the enter case, otherwise we are in the exit case. We can check for total internal reflection and if we don't, we can proceed and return values for wi.

Lastly, we can implement glassy materials as described in the spec. We first check if there is total internal reflection, and if it does then we reflect with pdf=1. Otherwise we do a coin flip to either reflect or refract.

Part 4

In this part of the project, we implemented a thin lens model for our camera. With the pinhole camera model, we assumed that light was only entering the camera through a tiny hole, so everything is in focus. However, most cameras (and the human eye) use the thin lens model. With the thin lens model, objects are in focus only if they're within a plane that is some distance from the lens (this distance depends on the focal length of the lens). This means that there will be a range of depth in the scene that will be in focus, and anything outside that range will be out of focus (if it is too close or too far back).

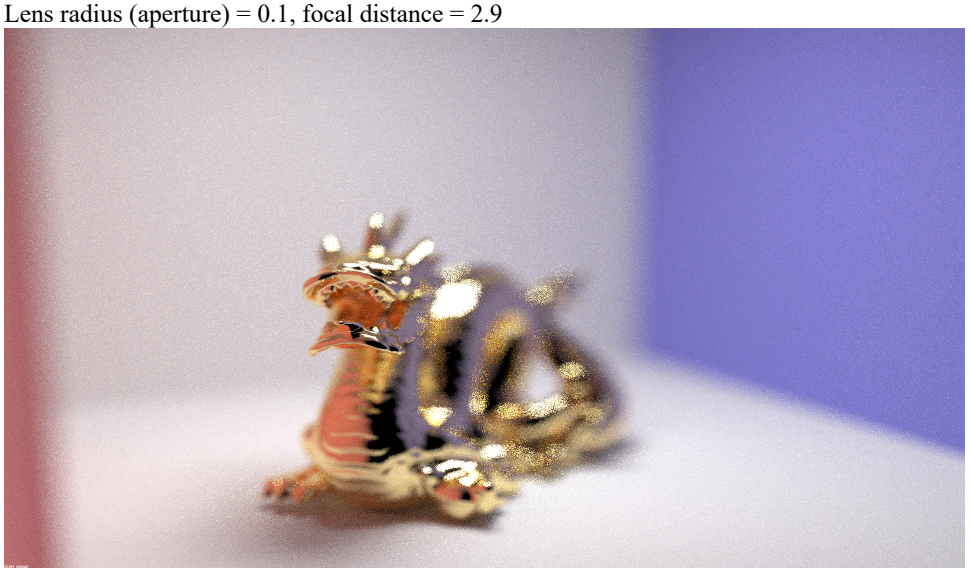
Implementation

We model the thin lens with a disk of given lens radius, and sample an intersection point of a ray from the image plane with a point on the lens. We then generate the intersection point of the refracted ray with the plane of focus, and added in the original position on the image plane to that ray. We return this modified ray instead of the ray directly from the image plane to the plane of focus.

Focus Stack

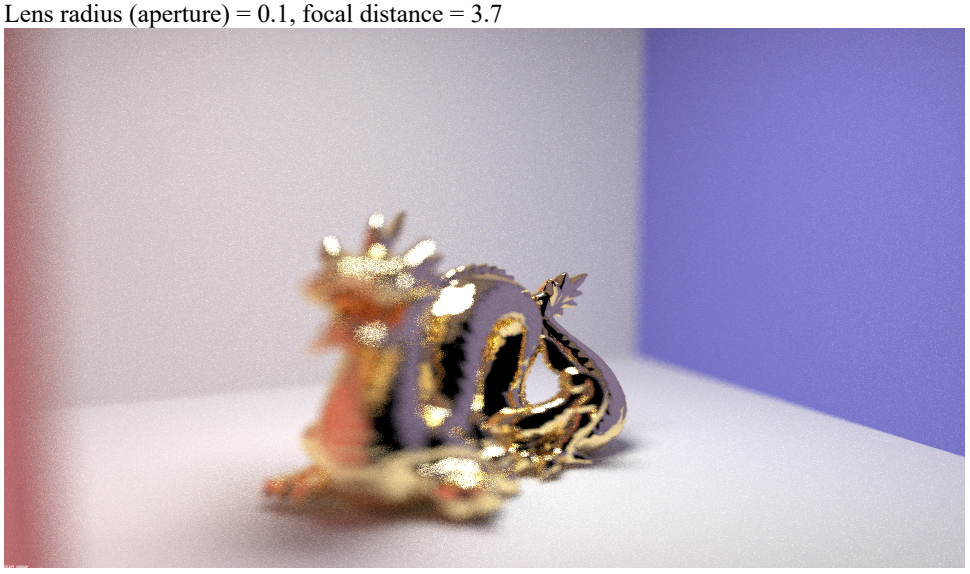
Here we can see the affects of varying the focal distance while keeping the aperture constant. As we increase the focal distance, we see that the depth of focus increases and the area in focus seems to move further and further "back" into the image. This is because increasing focal distance increases the sharpness of the viewing angle and the magnification of the image.

Lens radius (aperture) = 0.1, focal distance = 2.9

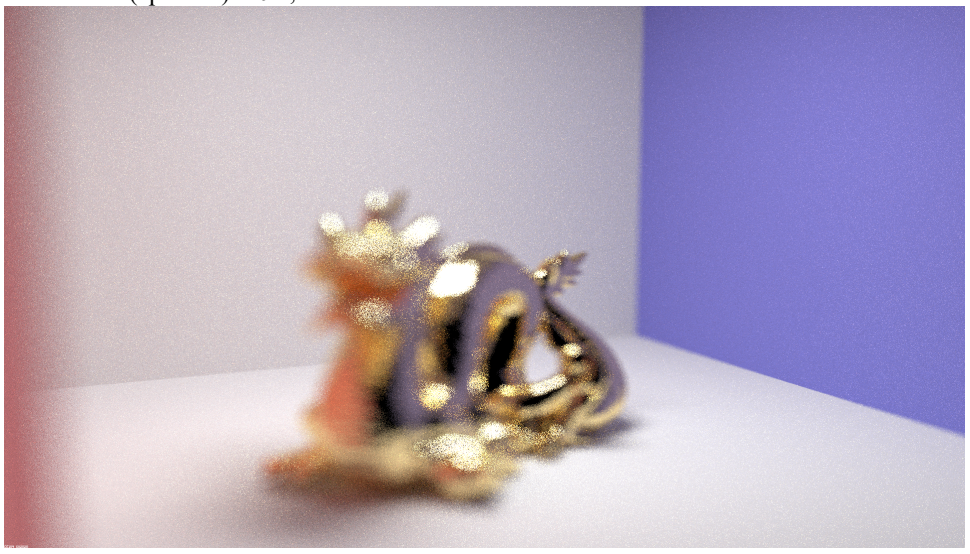


Lens radius (aperture) = 0.1, focal distance = 3.3

Lens radius (aperture) = 0.1, focal distance = 3.7



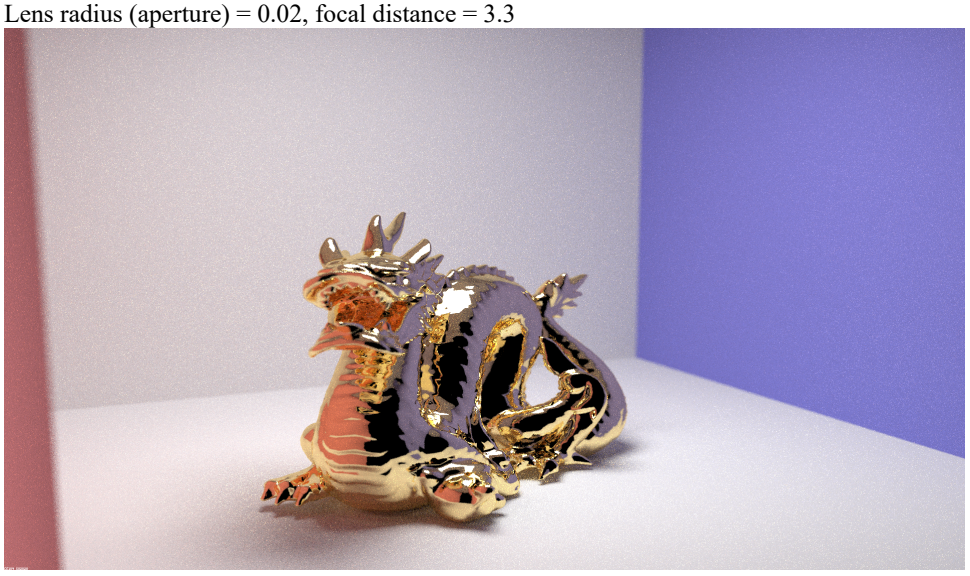
Lens radius (aperture) = 0.1, focal distance = 4.1



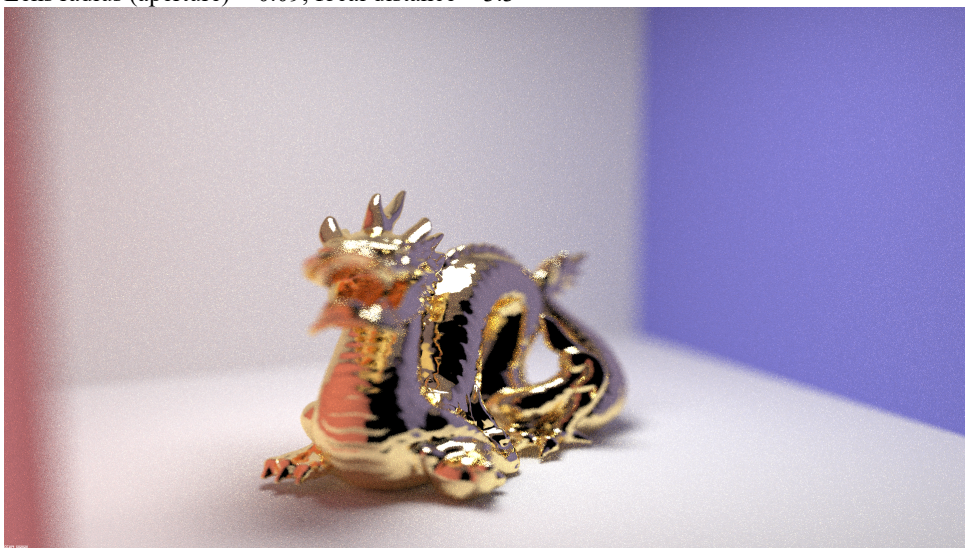
Aperture Stack

Here we can see the affects of varying the aperature while keeping the rendering focused on the center of the dragon. As lens radius (aperture) increases, the more light is let in to the camera, which means that depth of field decreases and the area in focus is smaller.

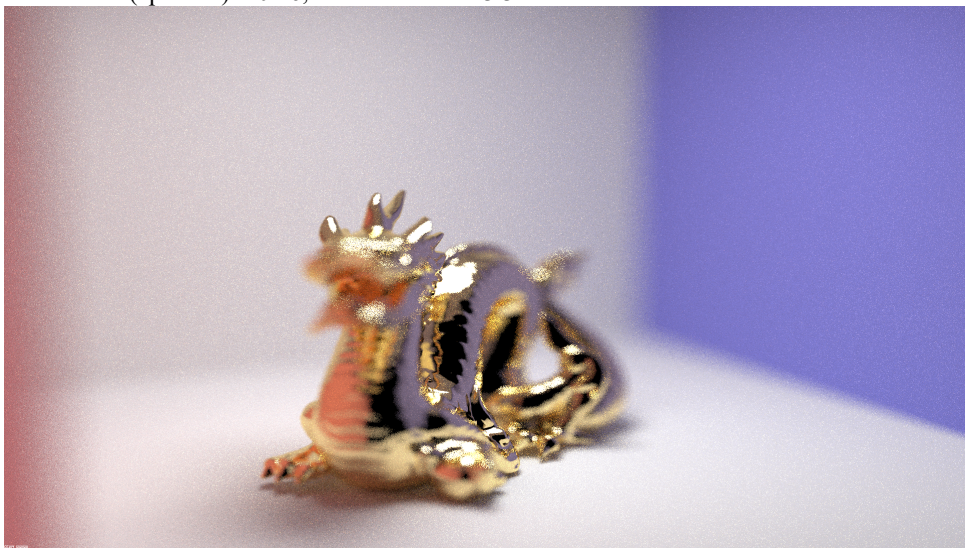
Lens radius (aperture) = 0.02, focal distance = 3.3



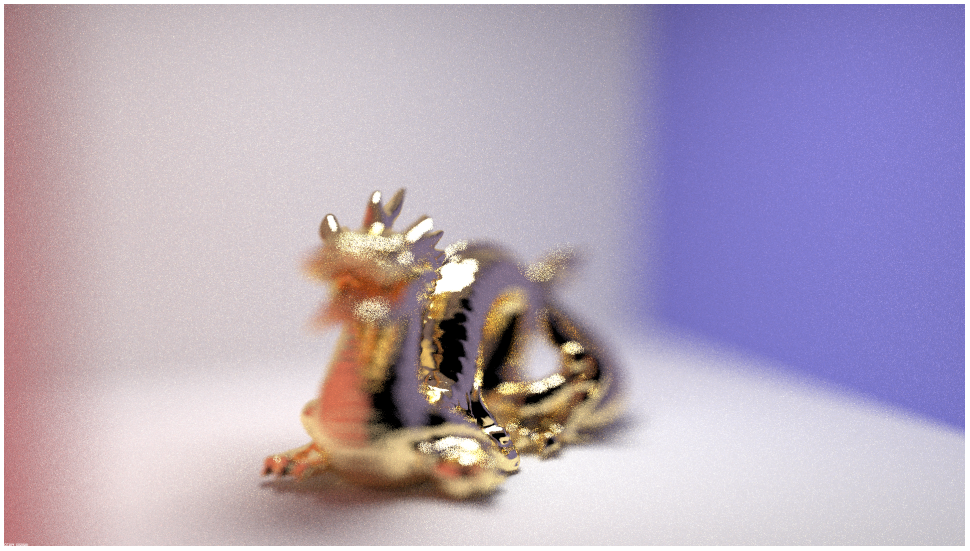
Lens radius (aperture) = 0.09, focal distance = 3.3



Lens radius (aperture) = 0.16, focal distance = 3.3



Lens radius (aperture) = 0.23, focal distance = 3.3



Collaboration

Overall, this was a fun project. We talked about the approaches for parts 1 and 4 and then each implemented one part on our own. For part 1, it was interesting to learn about Schlick's approximation/Fresnel Equations and how they can model a pretty accurate looking glass material. For part 4, it was cool to see how we could model real-world cameras and our own eyes using some fairly straightforward math to show the depth of field effect.