# Assignment 3-2: Additional Features to PathTracer

## Claire Liu, Justin Huey

## https://cal-cs184-student.github.io/project-webpages-sp23-JustinHuey1/

* NOTE: For this project, you will choose TWO out of the four given parts to complete. One of those parts must be Part 1 or Part 2. In other words, you can choose any combination of two parts except the pair (Part 3, Part 4).

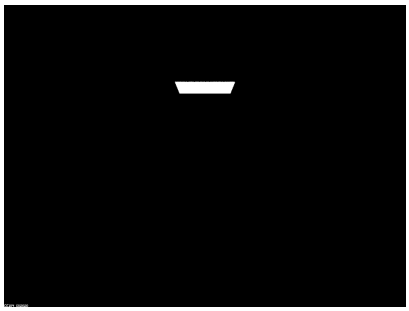
Overview: In this extension of the raytracing program, we are able to implement features that provide a more profound and realistic rendering of the scenes. Overall, we were able to explore with new techniques that are able to add new materials to the objects and give them a new look that is more realistic of the environment that they are in. In particular, we implemented mirror, glass, and microfacet material that gave very interesting looks for the spheres and dragon images that differed from the ones we were only able to render using the things we implemented in 3-1.

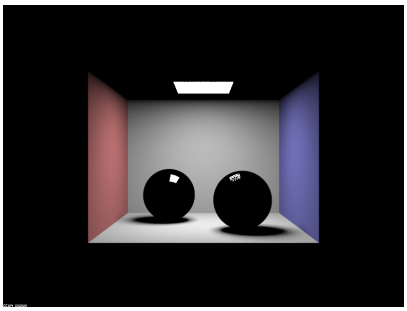
### Part 1. Mirror and Glass Materials

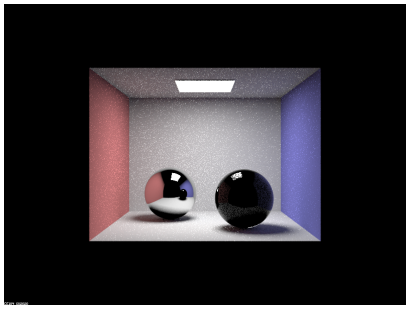
For this part, we implemented the mirror and glass material. We first worked on the reflection algorithm that reflected the vector across the z axis. This allowed us to implement the mirror material that reflects and returns the vector: reflectance / abs_cos_theta(*wi); We then worked on implementing refraction. We first figure out the value of eta by using the value of wo.z. If wo.z < 0, then eta = ior, otherwise eta is 1/ior. Then we check to see if we have total internal reflection. We can do this by checking whether 1 - (eta^2 * (1 - cos^2)) < 0. If this is true, then we have total internal reflection so our wi is empty and refraction doesn't occur. Otherwise, we let wi = Vector3D(-eta * wo.x, -eta * wo.y, oppSign * thetaCos) where thetaCos is sqrt(1 - (eta^2 * (1 - cos^2))) and oppSign makes sure that wi.z has the opposite sign of wo.z. To sample refraction, we first check if it refracts. If it doesn't refract, then we just return an empty vector, otherwise we return transmittance / abs_cos_theta(*wi) / pow(eta, 2). To sample the glass, it is a bit more convoluted. We similarly check the total internal reflection. If this is true, then we just reflect and return reflectance / abs_cos_theta(*wi). If this is false, then we know that reflection and refraction will occur, but we need to calculate the ratio of the reflection energy to the refraction energy. To do this, we use Schlick's approximation to decide the ratio. This is done by computing Schlick's reflection coefficient and passing that in the coin_flip as an argument to give us the probability of either reflecting or refracting. If the coin_flip is true, then we reflect, set the pdf to the reflection coefficient R, and return R * reflectance / abs_cos_theta(*wi). Otherwise we refract, so also set the pdf to (1-R), and (1-R) * transmittance / abs_cos_theta(*wi) / eta^2.

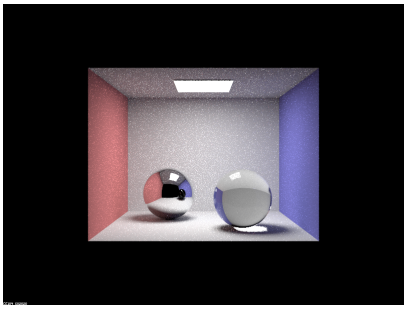
**Show a sequence of six images of scene `CBspheres.dae` rendered with `max_ray_depth` set to 0, 1, 2, 3, 4, 5, and 100. The other settings should be**

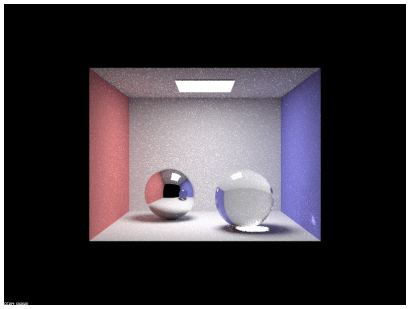
**at least 64 samples per pixel and 4 samples per light. Make sure to include all screenshots.**

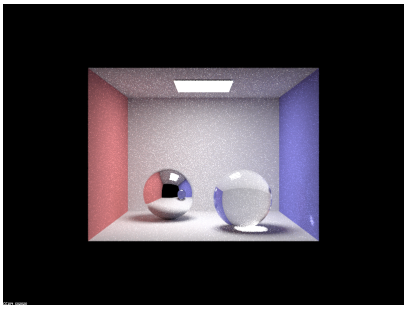

max ray depth: 0

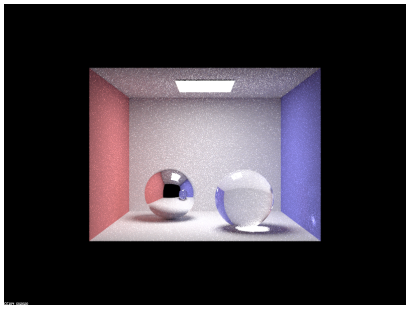

max ray depth: 1


max ray depth: 2


max ray depth: 3


max ray depth: 4


max ray depth: 5


max ray depth: 100

As we can see from the images, the more bounces that the rays can take, the better our images look. More bounces allow the ray to gather as much information as possible and get into all the cracks and shadows of all the objects in the scene, thus making our scenes look much more realistic. This also allowed our materials to truely show their distortions and how they work when put in the environment.

**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.**

The bounces effect the spheres differently since the left one is mirror and the right one is glass material.

For m = 0: Only emmission of the light source is shown, so just the light source is lit and everything else is black.

For m = 1: There is only direct lighting in the scene so the light source and any spots hit by the light source is lit. Due to the nature of the mirror material, the ray needs to bounce to the camera as well to show the color, so the left sphere is mostly black. This is mostly similar for the glass material.

For m = 2: There is partial color to the sphere on the left since there is the bounce from the sphere to the camera. The glass sphere is still dark since it requires an extra bounce. There are some speckles of white on the glass sphere. The shadows for the spheres are less dark and more smooth.

For m = 3: The images on the reflection is brighter for the mirror sphere. Also fixes the ligh for the ceiling to be the regular color rather than a darker representation. The color off the reflection in the mirror sphere of the glass sphere is black even thought the glass sphere is white. The reflections in the mirror sphere are sort of behind since they lack the one extra bounce to get to the camera to reflect the color represented in the scene. The glass sphere gains the color and also the white area that can be seen under it.

For m = 4: The color off the reflection in the mirror sphere of the glass sphere is finally matching up with the color, although still a bit dark. Overall, the scene is brighter since there are bounces to gain more information and make a more detailed scene. There is also the speck of white on the bottom of the glass sphere and the blue wall that is coming from the white area under it.

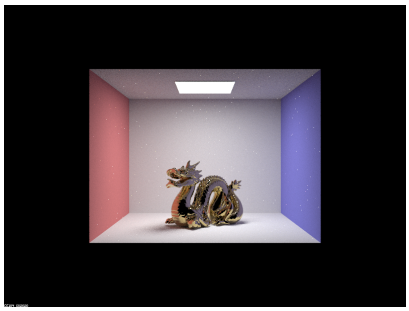
For m = 5: The scene is pretty much identical to the m = 4 scene, but a bit brighter.

For m = 100: The scene is much much brighter than when the max bounce depth was lower. The coloration and disruptiveness of the materials are really highlighted in these lighter environments. Also brings out the shadows and how smooth they are.

## Part 2. Microfacet Material

In this one, we implement the microfacet material. We first implemented the BRDF evaluation function using the fresnel term, shadow masking term, the normal distribution function, and the macro surface normal. We then had to implement the normal distribution function and fresnel term. For the NDF, we are defining how the microfacets' normals are distributions. To do this, we use Beckmann distribution. For the fresnel term, we are calculating the air conductor fresnel term using the approximation. Lastly, we implement BRDF importance sampling.

**Show a screenshot sequence of 4 images of scene `CBdragon_microfacet_au.dae` rendered with $\alpha$ set to 0.005, 0.05, 0.25 and 0.5. The other settings should be at least 128 samples per pixel and 1 samples per light. The number of bounces should be at least 5. Describe the differences between different images. Note that, to change the $\alpha$, just open the .dae file and search for `microfacet`.**


alpha: 0.005

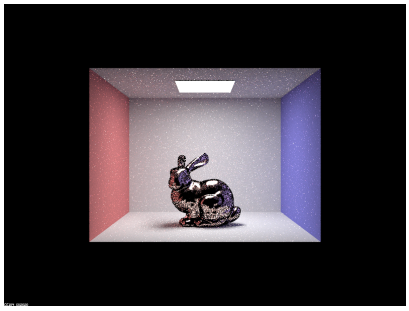

alpha: 0.05

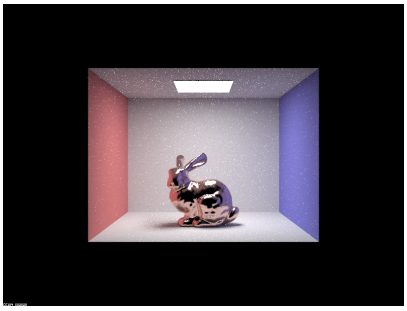

alpha: 0.25


alpha: 0.5

As we can see from the images, alpha can make the model vary in how its seen. Alpha specifically dictates the roughness of the macro surface. The smaller alpha is, the smoother the macro surface is so the model looks more glossy and has the reflection to it. This is contrasted to when the alpha value is large and the model looks a bit brighter and has a more diffused look.

**Show two images of scene `CBbunny_microfacet_cu.dae` rendered using cosine hemisphere sampling (default) and your importance sampling. The sampling rate should be fixed at 64 samples per pixel and 1 samples per light. The number of bounces should be at least 5. Briefly discuss their difference.**

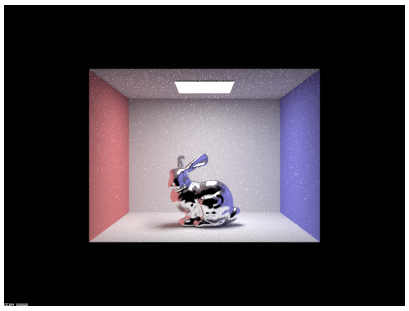

Hemisphere sampling


Importance sampling

As we can see from the images, the hemisphere sampling is a bit darker compared to the importance sampling. The darkness is due to the increased noise that is prevalent by using hemisphere sampling. Importance sampling will only use samples that intersect with the light source so the samples are guaranteed to add

valuable information to the overall image. Therefore making the image look more defined and the material easier to see. The hemisphere sampling is not as good here since we are using Beckmann distribution and not diffuse BRDFs.

**Show at least one image with some other conductor material, replacing `eta` and `k`. Note that you should look up values for real data rather than modifying them arbitrarily. Tell us what kind of material your parameters correspond to.**



Simple inorganic materials: n = 0.16927 k = 12.327

The different kinds of materials make very different interpretations of the mode. The materials can provide a different color and feeling that affect how the view sees the bunny. The simply inorganic material makes the bunny have a silver and shinier look to it.

## Part 3. Environment Lightl

**Pick one *.exr* file to use for all subparts here. Include a converted *.jpg* of it in your website so we know what map you are using.**

**In a few sentences, explain the ideas behind environment lighting (i.e. why we do it/how it works).**

Your response goes here.

**Show the *probability_debug.png* file for the *.exr* file you are using, generated using the `save_probability_debug()` helper function after initializing your probability distributions.**

Your response goes here.

**Use the `bunny_unlit.dae` scene and your environment map *.exr* file and render two pictures, one with uniform sampling and one with importance sampling. Use 4 samples per pixel and 64 samples per light in each. Compare noise levels. Make sure to include all screenshots.**

Your response goes here.

**Use a different image (if you did part 2, we recommend `bunny_microfacet_cu_unlit.dae`) and your environment map *.exr* file and render two pictures, one with uniform sampling and one with importance sampling. Use 4 samples per pixel and 64 samples per light in each. Compare noise levels. Make sure to include all screenshots.**

Your response goes here.

## Part 4. Depth of Field

**For these subparts, we recommend using a microfacet BSDF scene to show off the cool out of focus effects you can get with depth of field!**

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

Your response goes here.

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

Your response goes here.

**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.**

Your response goes here.

### Working with partner

We worked through all the parts together and went through them one at a time. We mainly pair programmed and bounced ideas off each other. We tried a lot of different things and explored different ideas about how to approach these questions and solve them. We learned a lot about the things we learned in lecture and were excited to apply the things we learned into a working program and be able to see how they worked.