CS184: Computer Graphics

Sping 2022

Assignment 3-2: Pathtracer

Prince Wang

Overview

An overview of the project, your approach to and implementation for each of the parts, and what problems you encountered and how you solved them. Strive for clarity and succinctness. On each part, make sure to include the results described in the corresponding Deliverables section in addition to your explanation. If you failed to generate any results correctly, provide a brief explanation of why. The final (optional) part for the art competition is where you have the opportunity to be creative and individual, so be sure to provide a good description of what you were going for and how you implemented it. Clearly indicate any extra credit items you completed, and provide a thorough explanation and illustration for each of them.

Part 1: Mirror and Glass Materials

Requirements

- Make sure you include a short but clear summary of your implementation
- 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
- 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 you include a short but clear summary of your implementation:

There are four functions that I implemented in total for this part.

- BSDF::reflect()

For reflection, we simply use the normal of the surface to determine the reflected ray's coordinate. This is given by In this case our normal is Vector3D(0,0,1), so it's a one liner.

- MirrorBSDF::sample_f()

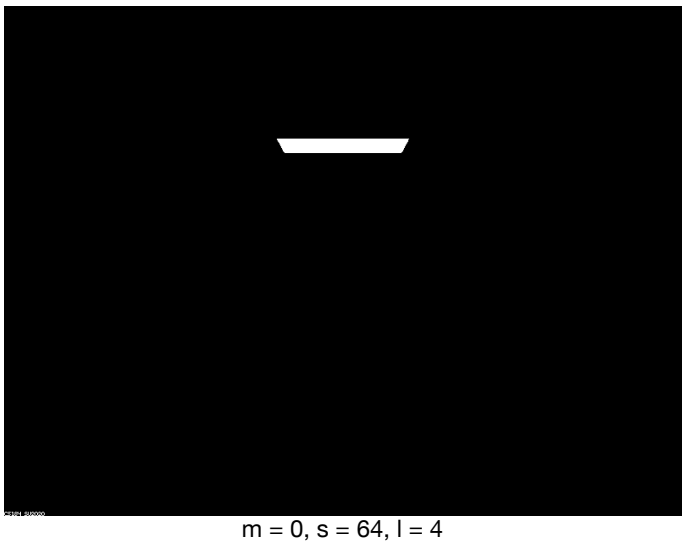
This one is also straight forward. We first call the reflect() function, then we set the pdf to 1. At last we return (this->reflectance) / abs_cos_theta(wi). Notice that we need to run the case when max_depth is >= 1 because in order to see light reflection, we need to bound the light off the object into the camera. When max_depth = 0, we only see the light from light source.

- BSDF::refract()

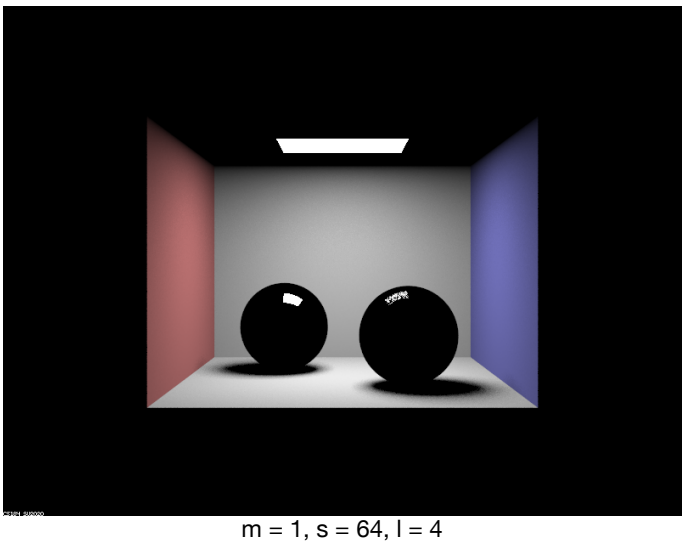
This function is a little harder to implement. We need to calculate the correct change in ray's direction. The majority of the algorithm is given by the spec. The key is to understand how ior changes when entering and exiting the material. Another important detail is to check if we have total internal reflection, where we simply return false and exit the function. This check can be done by checking if 1.0 - pow(ior, 2) * (1.0 - pow(cos_theta, 2)) is < 0.

- MicrofacetBSDF::sample_f() I coded up the algorithm according to the pseudo-code given by the spec. This function is not difficult, the key is to check for internal reflection. It there is internal reflection, assign reflection of wo to wi, set pdf to 1, and return reflectance / abs_cos_theta(wi). Otherwise, we doa. coin flip to decide whether to do reflection or refraction.

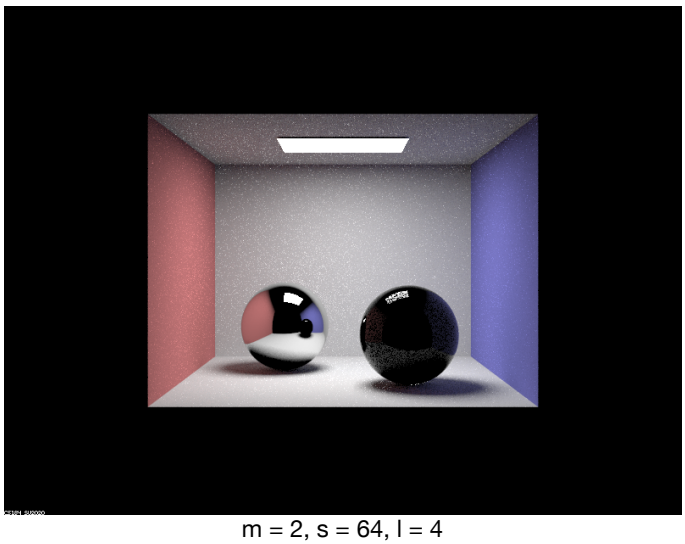
Show a sequence of six images of scene CBspheres.dae rendered with max_ray_depth set to 0, 1, 2, 3, 4, 5, and 100. The other settings should be at least 64 samples per pixel and 4 samples per light



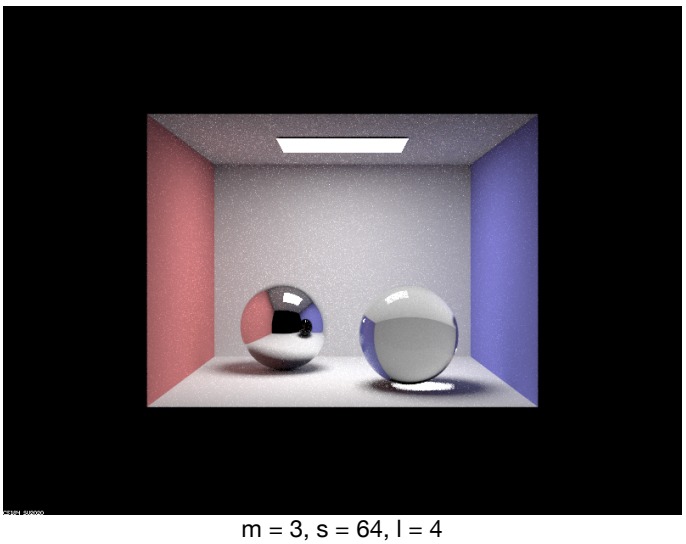
m = 0, s = 64, l = 4



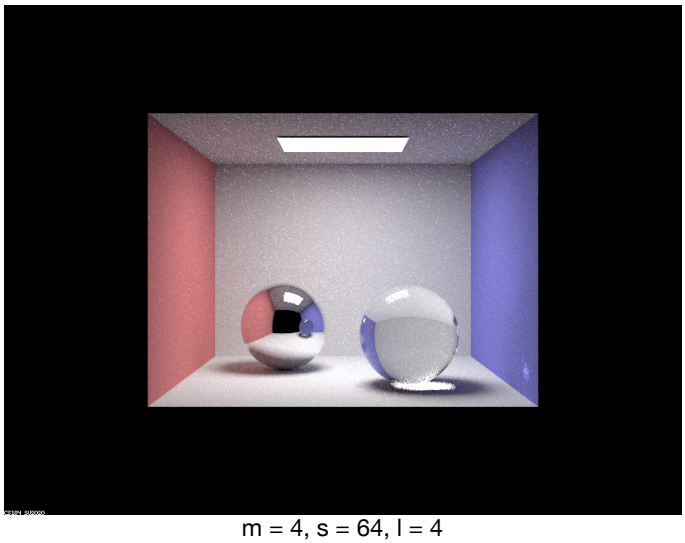
m = 1, s = 64, l = 4



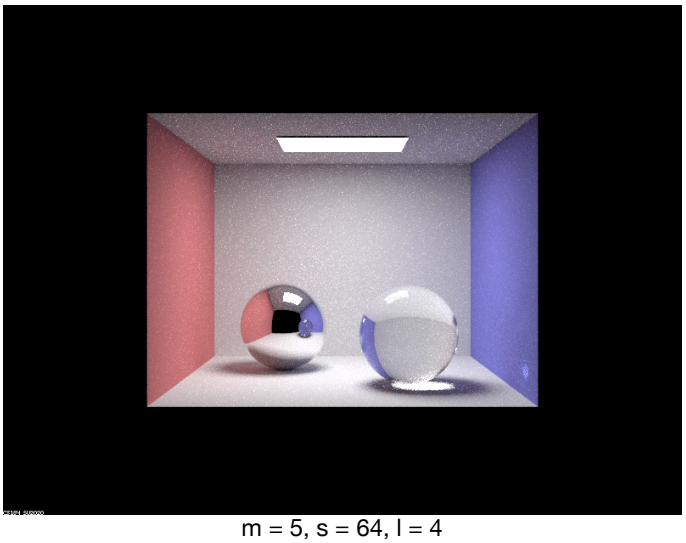
m = 2, s = 64, l = 4



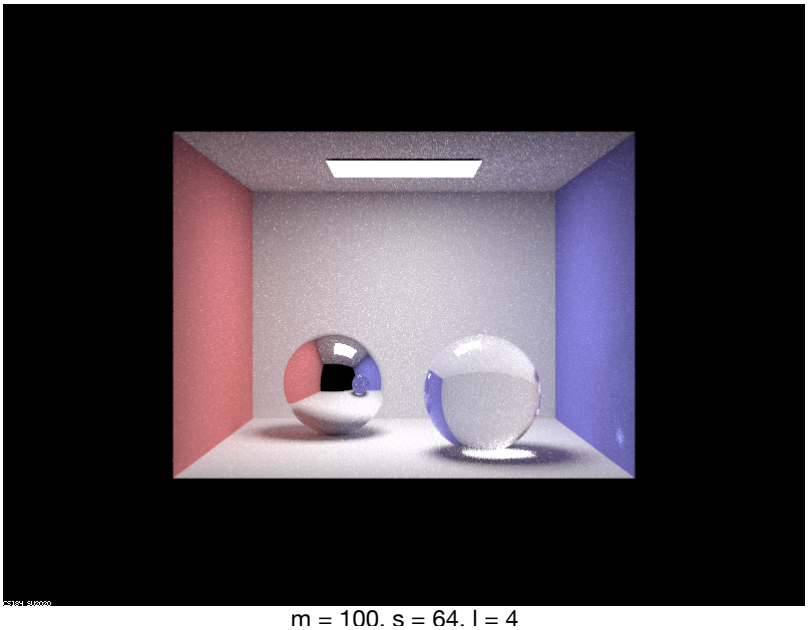
m = 3, s = 64, l = 4



m = 4, s = 64, l = 4



m = 5, s = 64, l = 4



m = 100, s = 64, l = 4

Point out the new multibounce effects that appear in each image and explain how these bounce numbers relate to the particular effects that appear

It is very interesting to compare the multibounce effect. I realized that only for higher number of max_ray_depth can we see a ray going though the material, coming out, getting bounced off from a wall and reentering the material again. This is not happening for low bounce numbers.

For m = 0, we only see light from light source as there is no ray bouncing off the object and entering the camera. With m = 1, we can already see the room, as light bounces off from the wall and entered the camera. The two spheres are interesting, we can see a portion of light hitting the sphere and bouncing off into the camera, but the majority of it is dark, as our max_ray_depth does not allow light coming off the glass material yet. With m=2, we can now see the reflected spheres, but not the refracted sphere. We begin to see the refracted sphere at m=3, as most light at that sphere bounce on the surface, go through it, and exit toward the camera, needing 3 bounces. At m = 4, we see the reflection of the right sphere inside the left sphere, as we have allowed one more bounce. Beyond m=5, we don't see much difference.

Part 2

Requirements

- Show a sequence of 4 images of scene CBdragon_microfacet_au.dae rendered with \alphaα 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 \alphaα, just open the .dae file and search for microfacet.
- 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.
- 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

Make sure you include a short but clear summary of your implementation:

For this part I implemented four functions.

- MicrofacetBSDF::f() This function is quite straight-forward, I implemented the formula to calculate f on the spec. I also add a check at the beginning of the function. If the dot product between our wo/wi and the normal vector is negative, we return 0 and exit early.
- MicrofacetBSDF::D() This function is also straight-forward. The formula is a little bit complicated then the last one but not very long. I coded it up in a few lines.
- MicrofacetBSDF::F() This function is also straight-forward. I coded it up in a few lines according to the formula on the spec.
- MicrofacetBSDF::sample_f() This function is the most challenging among all 4 parts. For this function, I first get a random sample from the sampler. Then I calculate theta_h, phi_h and use them to determine the new vector h. Using h I calculated the new angle wi. If the dot product between normal vector and wi is non-negative, we process and calculate the pdf of sampling h with respect to the solid angle. Then we find the final pdf of sampling and return it.

Show a sequence of 4 images of scene CBdragon_microfacet_au.dae rendered with \alphaα 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 \alphaα, just open the .dae file and search for microfacet.

alpha = 0.005

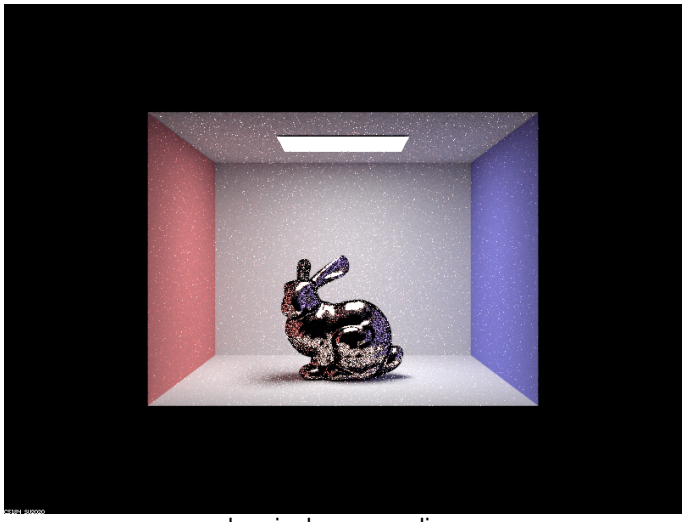
alpha = 0.05

alpha = 0.25

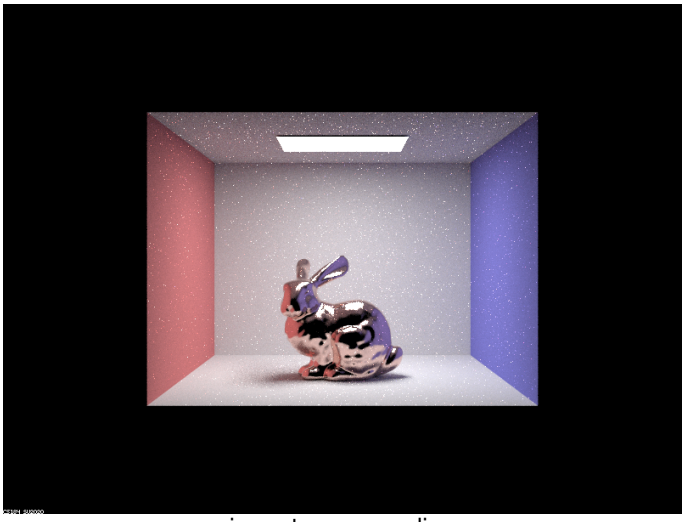
alpha = 0.5

Notice that as alpha gets lower, the dragon looks more and more reflective and glossy. This is because that alpha is the roughness of the surface, and that the less rough it is, the more smooth it is. Our dragon with very low alpha will look very smooth and glossy. With high alpha, the dragon is less reflective.

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



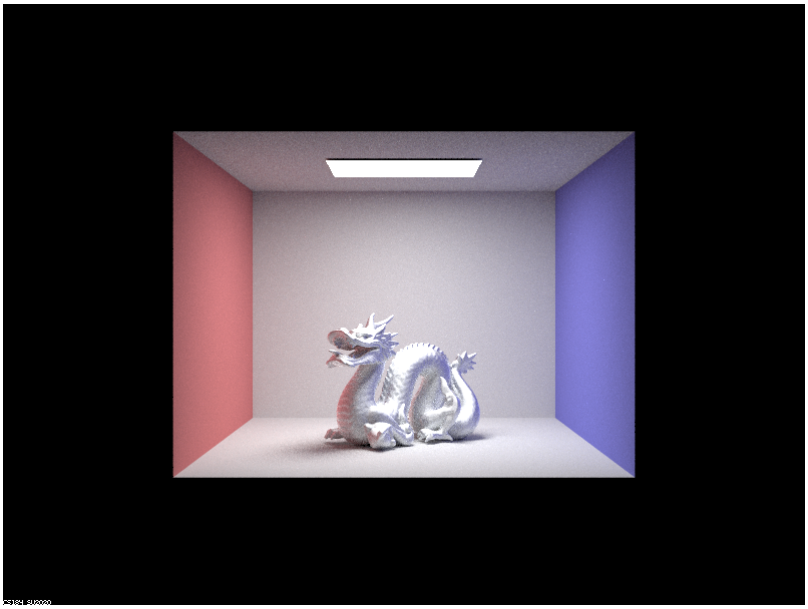
importance sampling

As we can see. the bunny with hemisphere sampling has a lot of parts that has holes in it. It looks a little noisy. On the other hand, the bunny with importance sampling looks very glossy, and is not noisy at all. This is due to the fact that under hemisphere sampling, some areas of the bunny does not have light reflect into the camera, as our sampling is not from a set area but from the whole hemisphere.

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

</tr></table> This material corresponds to lead. Parameters:

- s = 128
- l = 1
- m = 5
- eta = [1.0132, 0.65850, 0.42092]
- k = [6.2285, 5.6248, 4.6673]



dragon of lead material