Assignment 3: PathTracer

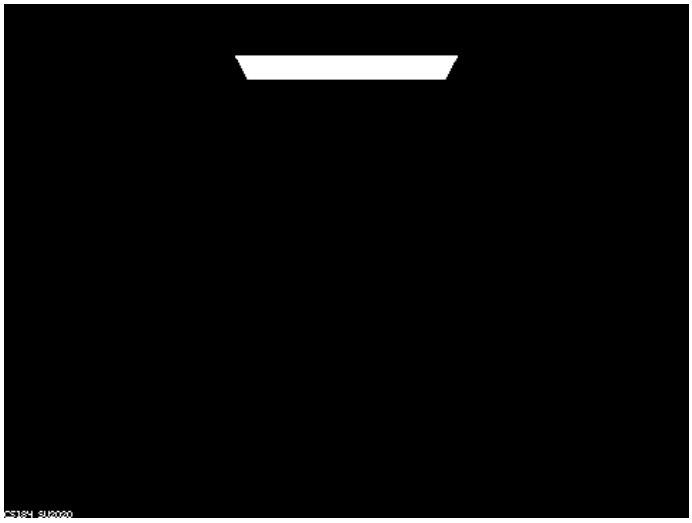
Yang Huang, Juntao Peng

In assignment 3-2 we first implemented glass and mirror material in part 1. Then we chose part 3 as out second part where we implemented environment light using uniform sphere sampling and inversion-based importance sampling.

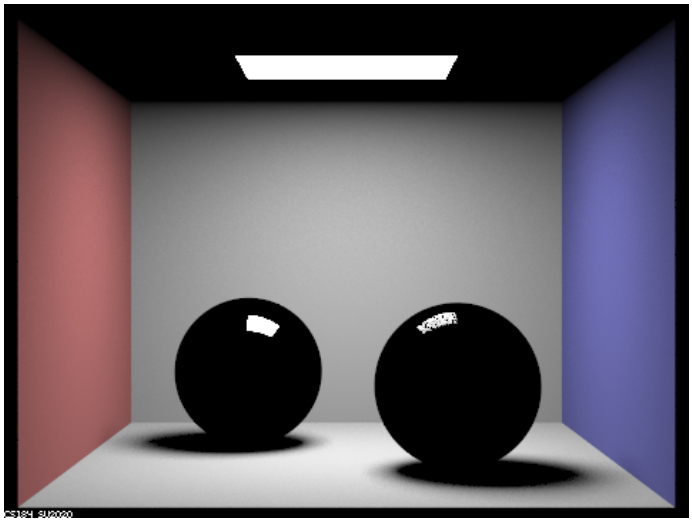
Part 1: Mirror and Glass Materials

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..

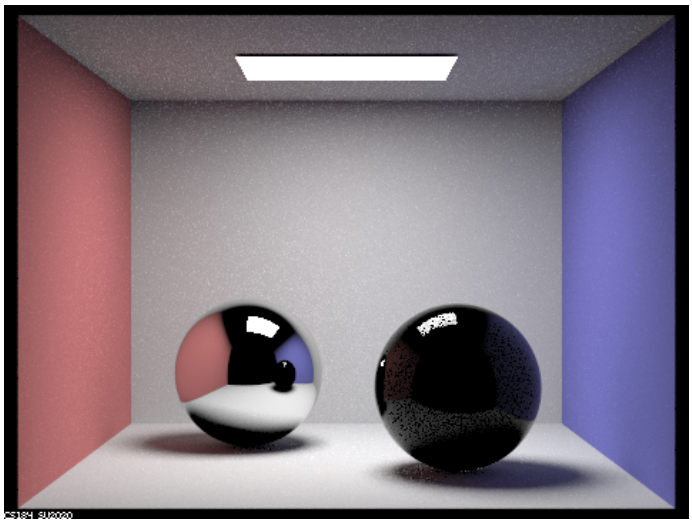
Point out the new multibounce effects that appear in each image. When m = 0, we can only see the cornell box since no bounce so we can only see light from area light.
When m = 1, there is one bounce from the light reflecting off the spheres and towards the camera.
When m = 2, we see a light that bounces off the wall into the left sphere that is then bounced into the camera. We can also notice a bright ceiling in image but a dark ceiling in the reflection of left sphere. Also, we start to see some reflected light in the right sphere but the shadow is still pure black.
When m = 3, we start to see the ceiling in the left sphere due to light bouncing off the floor and wall back to the ceiling and off the left sphere into the camera. However, the reflection of right sphere in left sphere is still black.
When m = 4, we can see a realistic reflection of right sphere in left sphere and also notice a small transparent shadow in right wall.
From m = 5 to 100, we can see a more and more realistic reflection of mirror and glass material. We can gradually see the reflection of right sphere in left sphere, see a transparent shadow on right wall, see a smooth shadow caused by refraction in right sphere.



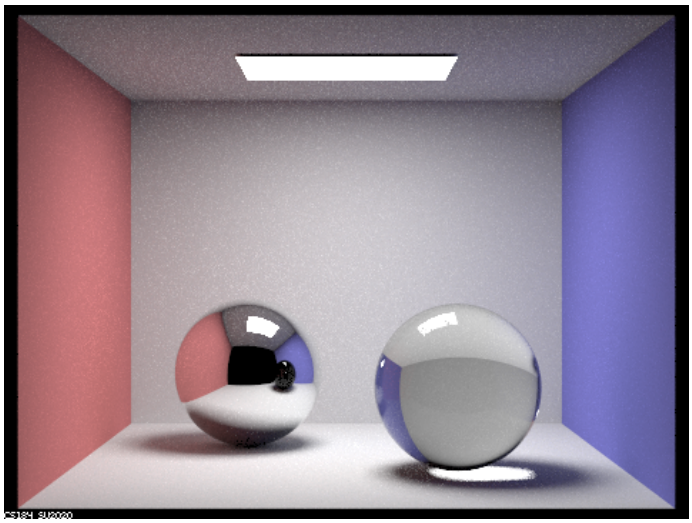
max_ray_depth = 0, 256 samples per pixel, 4 samples per light



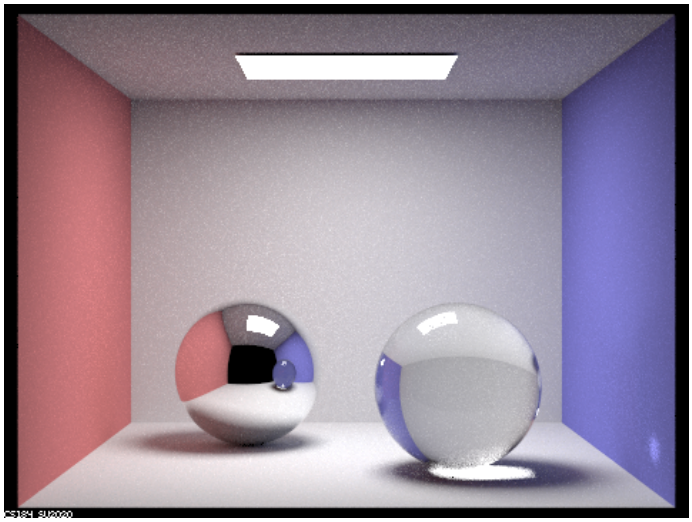
max_ray_depth = 0, 256 samples per pixel, 4 samples per light



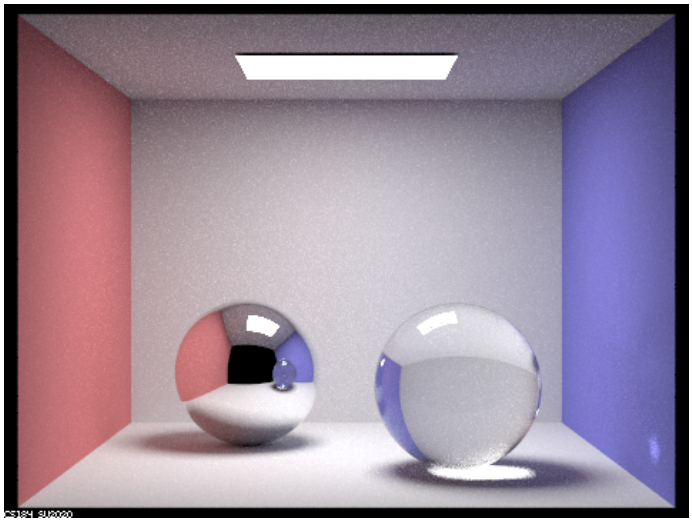
max_ray_depth = 2, 256 samples per pixel, 4 samples per light



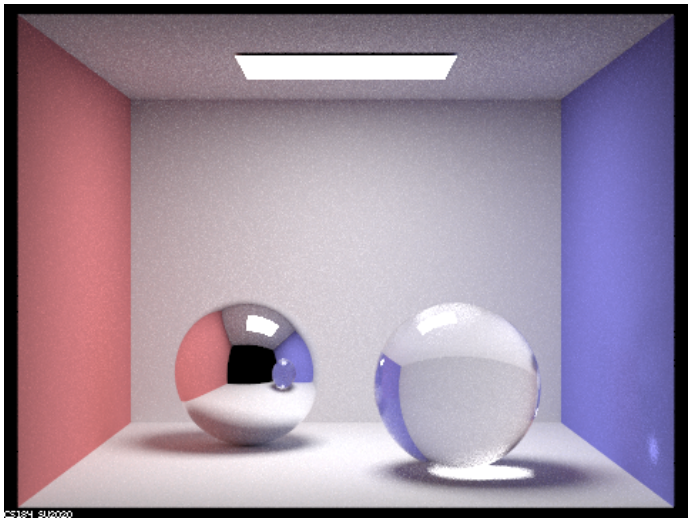
max_ray_depth = 3, 256 samples per pixel, 4 samples per light



max_ray_depth = 4, 256 samples per pixel, 4 samples per light



max_ray_depth = 5, 256 samples per pixel, 4 samples per light



max_ray_depth = 100, 256 samples per pixel, 4 samples per light

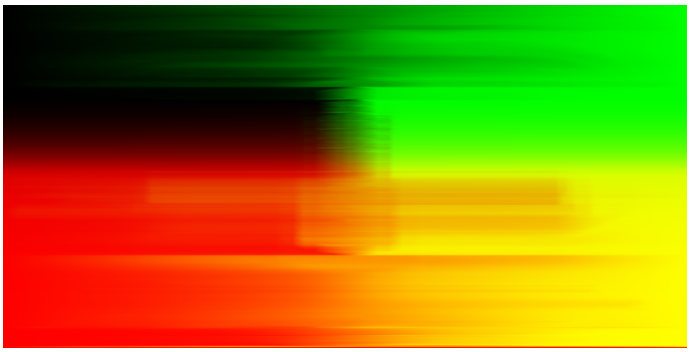
Part 3: Environment Light

In part 3 we first write a `sample_dir` function with which we can trace along the direction of a ray until we find the meet point on the environment map. This is achieved by converting global direction to theta-phi representation and then convert the angle to environment map using a lerp operation.

The second task of part 3 is uniform sphere sampling. We write `sample_L` function with which we can sample a uniformly random direction from a hitpoint and find the corresponding xy value on the environment map.

The third task of part 3 is importance sampling using inversion method. We first update the `init` function by storing `pdf_envmap` with the environment map's pixel value and then normalize it. Then we compute the marginal cumulative density function in `marginal_y`, which is a row-wise summation of `pdf_envmap` along the width axis. The last step in the `init` function is to compute `cond_y` by dividing values in `pdf_envmap` by the neighbor difference of `marginal_y` (this will give us the true probability of P(y)).

After finishing `init` function, we need to update `sample_L` with importance sampling.
1. We first perform a 2D uniform sampling from [0, 1).
2. Then use the value of `sample[1]` to inversely importance sample from the marginal distribution of y. This is achieved by using `std::upper_bound` to compute the index of the next larger cumulative probability density.
3. After getting a fixed y-value, we repeat the inversion sampling on the conditional distribution of x given y to get x-value.
4. Once we get the xy combination, we lerp it on the environment map, and convert it to the world direction.
5. Finally we set the value of pdf pointer to P(y) * P(xly) and then multiply it by `w * h / (2 * PI * PI * sin(theta))`.



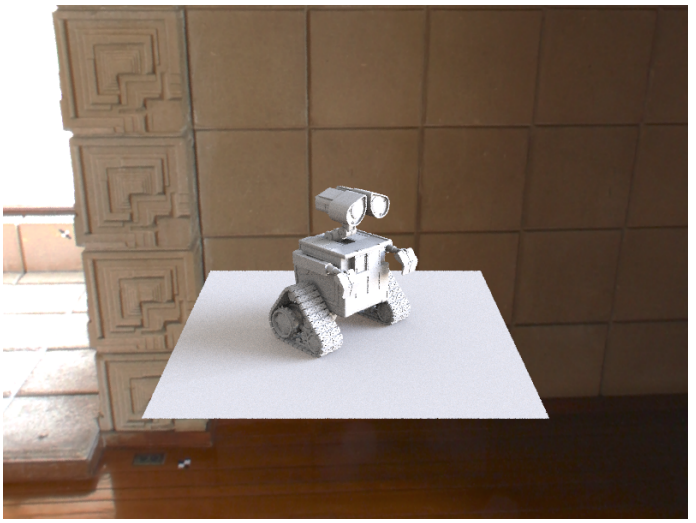
Importance Density of ennis.exr

Bunny using uniform sphere sampling

Bunny using importance sampling

When rendering our bunny the noise in uniform sphere sampling is higher than the one in importance sampling. We can see black dots on the surface of the left bunny.

Wall-e using uniform sphere sampling



Wall-e using importance sampling

When rendering our Wall-e the noise level between these two methods are closer than that of the bunny's. This is probabily because there is more polygons in the dragon mesh so that the ray samples are more likely to hit the model and thus gives back a valid radiance from the environment light source.