Project 3 Global Illumination

Introduction

I implemented **pathtracing** and **backwards raytracing** (light raytracing) using Python and some utilities libraries.
To run the program install the requirements following the readme and then run:

```
$ python example_pathtracer.py
```

Light Raytracing

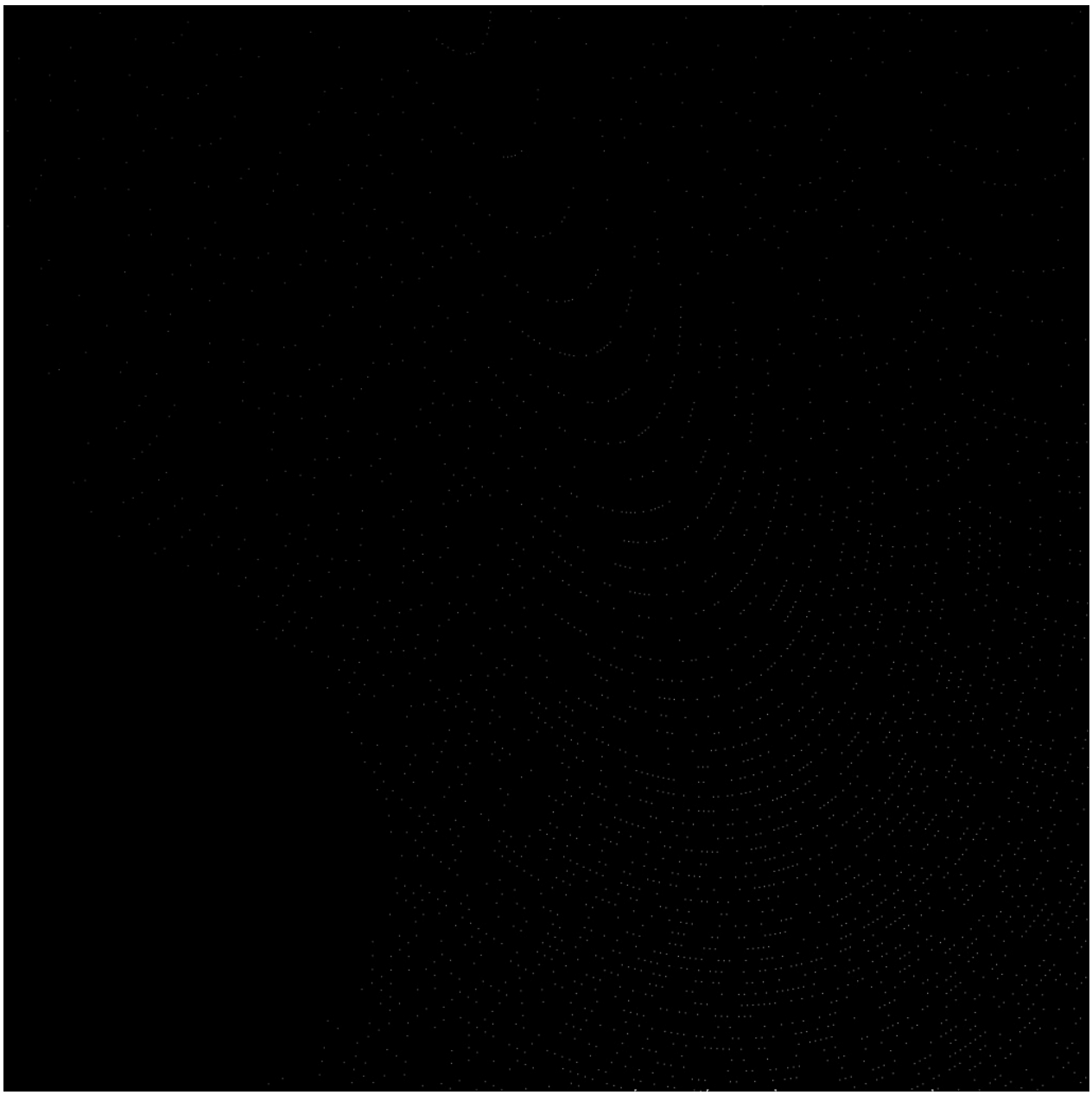
I followed the paper called Backwards Raytracing from 1986. I have a point light in the ceiling and rays are shot downwards to an imaginary plane that covers the floor of the room.

I created a transport_light function which takes a ray, scene, light intensity and depth. So the ray is intersected with the objects of the scene.
At the min hit point:

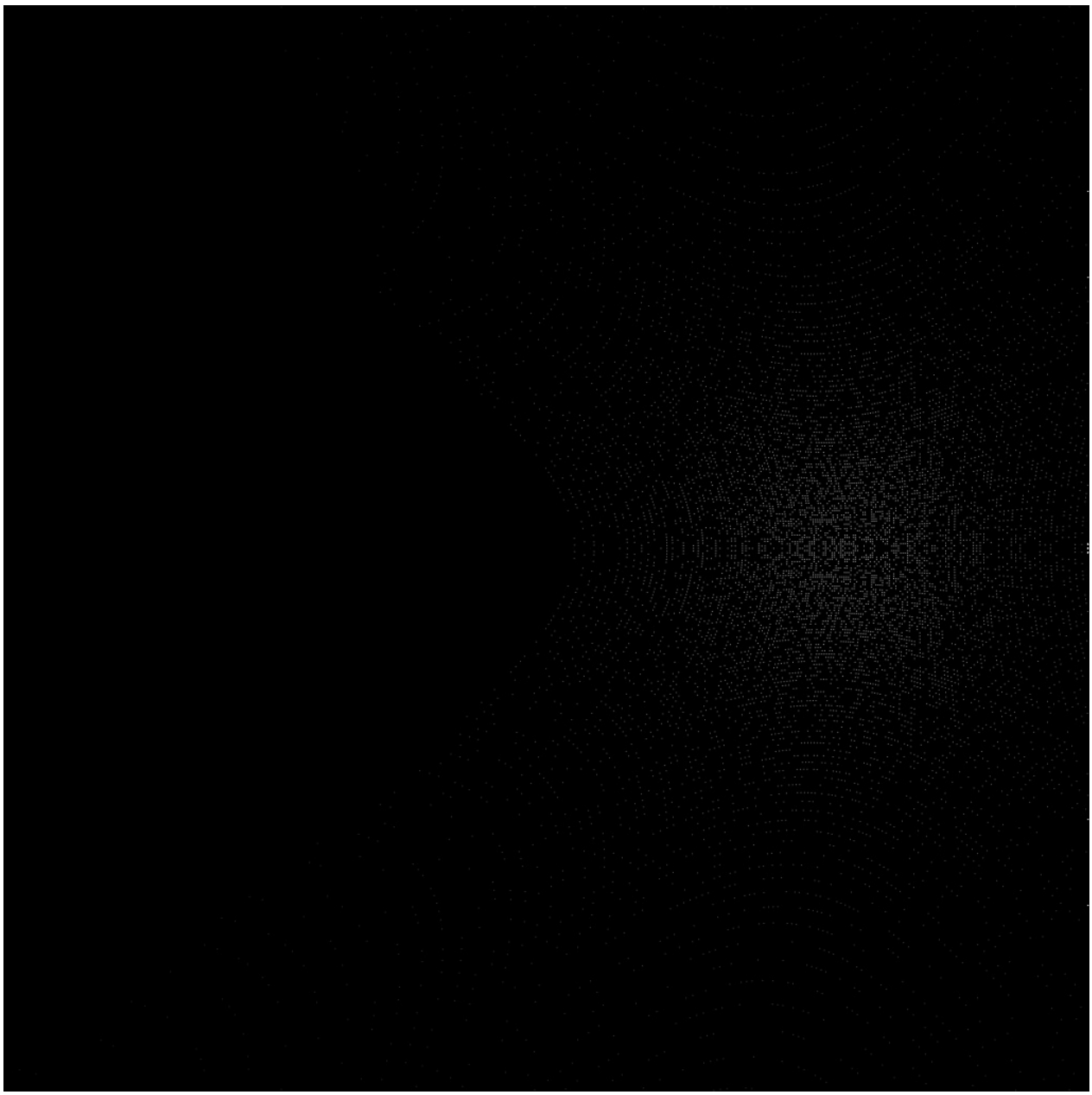
- if the surface is diffuse
  - and is the first recursion level stop
  - if the depth is more than 1, store the intensity in the light map at the uv for that ph (hit point)
- if it's reflective/specular bounce and shoot another ray with transport_light

Examples of Light Maps

This are some of the generated light maps. Resolution 1024x1024.



This is the 4th plane



This is the 5th plane

Path Tracing

I'm using paths with a sort of Russian Roulette, I followed some of the instructions from the rendering class of prof. Ramamoorthi

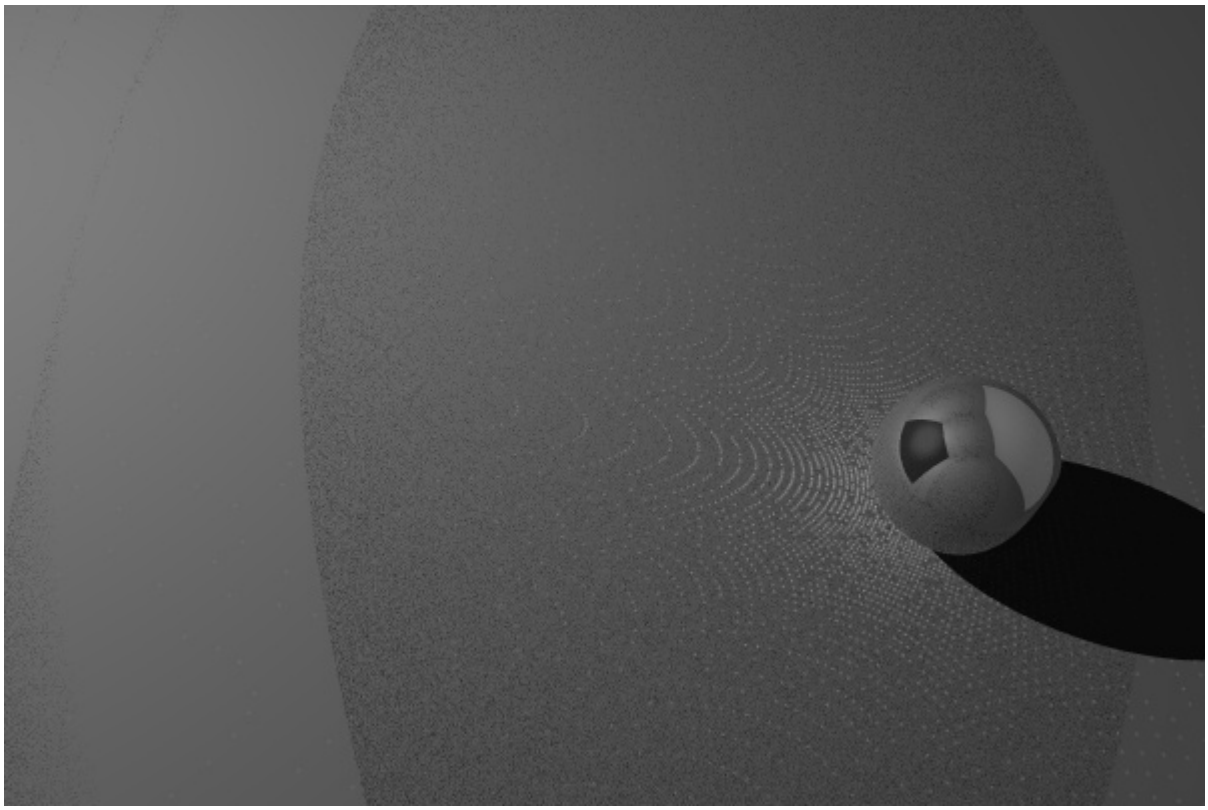
https://cseweb.ucsd.edu/~viscomp/classes/cse168/sp20/schedule.html

1. First I get the surface color for the hit point using the light and checking for occlusion.
2. I have a weight parameter in the recursive pathtrace function which updates in each call, by multiplying the *kr* of the surface. This means that the weight will be decreasing with each bounce.
3. Play Russian Roulette. If the random number is less than `1 - weight` stop. That means that with each bounce the probability of stopping increases. If this is the bounce number 5 stop no matter what, I did this because other projects where doing the same and it had good results.
4. If continue, reflect the ray and call pathtrace with that ray. If the surface is reflective the reflected ray only gets a little of variation, if the surface is diffuse the reflected ray is a random ray in the hemisphere.
5. For gathering the color samples of at the moment of calling pathtrace in the reflected ray this equation is applied.

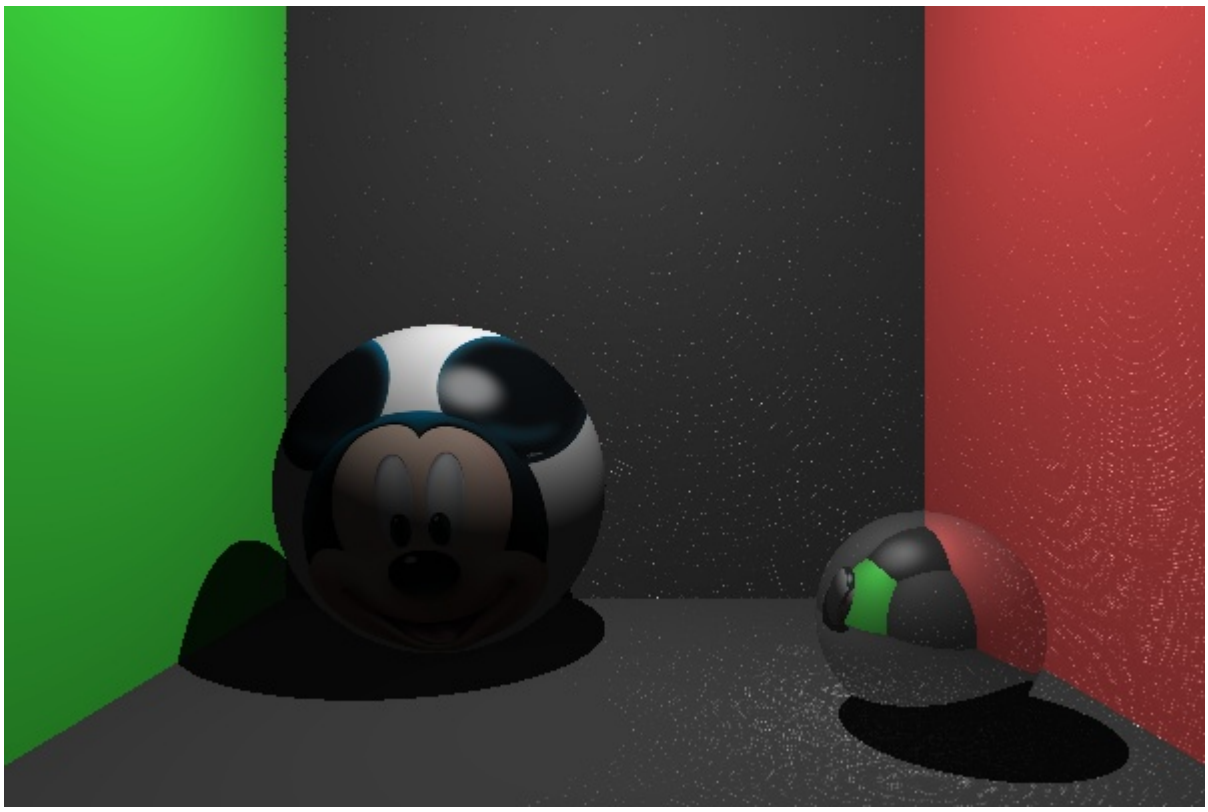
```
final_color = (1 - kr) * surface_color + kr * pathtrace(
    reflected_ray, scene, depth, current_level + 1, weight
)
```

Results

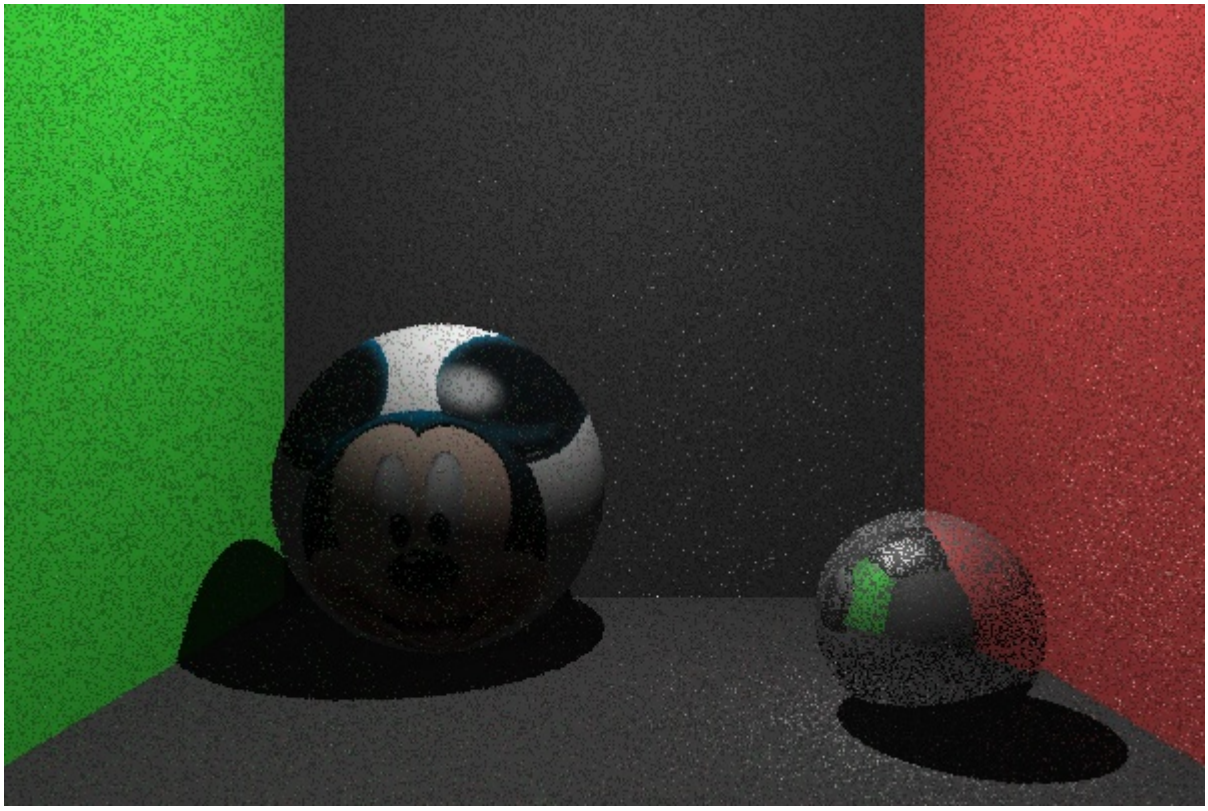
Images where rendered in resolution 600x400px.



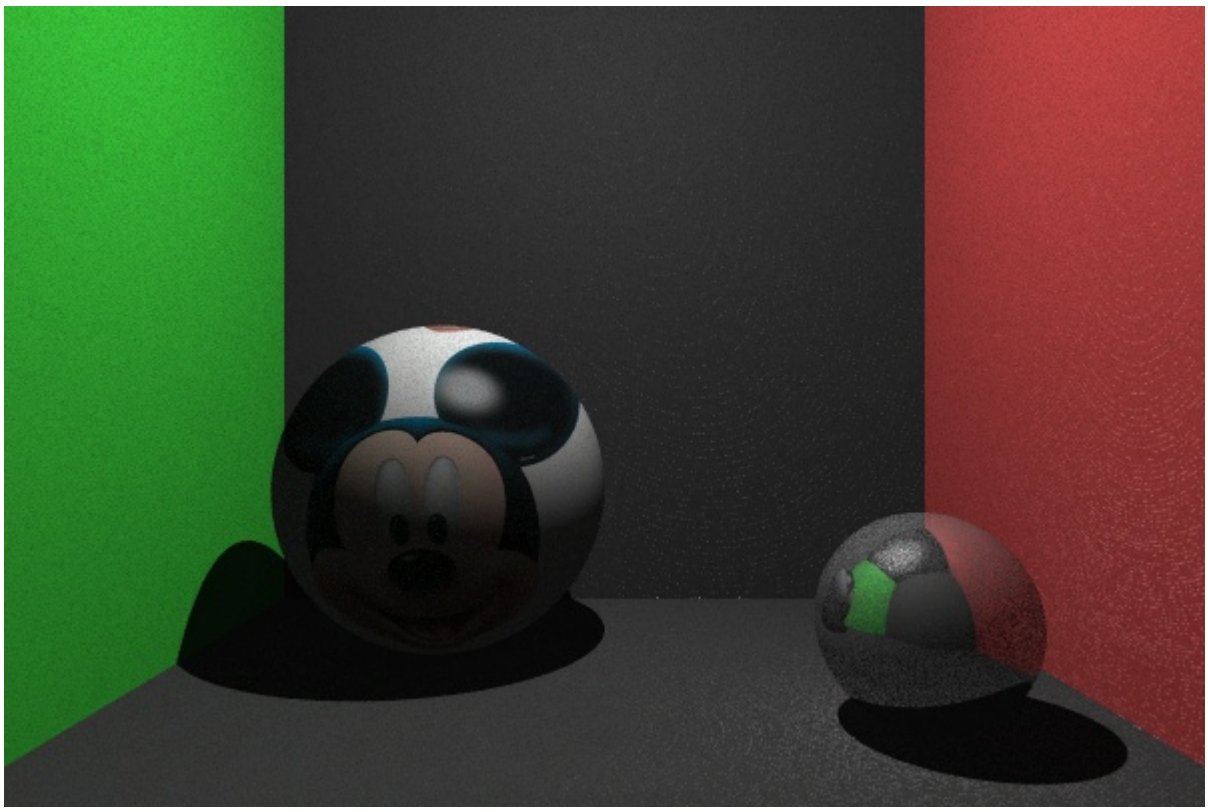
Light raytracing takes about 1 min, this was rendered with raytracing



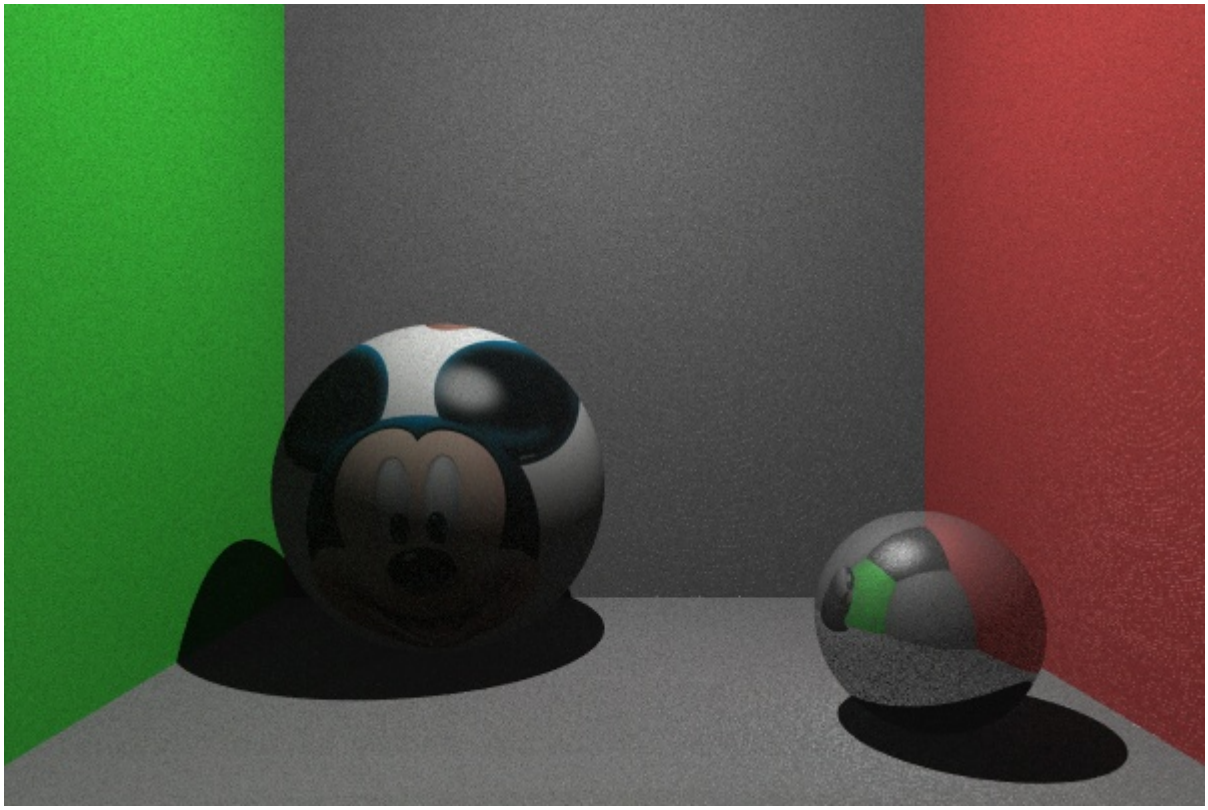
Raytracing takes about 1 min without anti-aliasing



Path Tracing with 1 path per pixel takes about 3 min



Path Tracing using 9 paths per pixel takes about 16 min it seems like using kr=0.3 and deciding to only give that amount of contribution to bounces is not enough to see color bleeding.



Path Tracing using 16 paths per pixel, I changed the planes to be lighter and kr for diffuse to 0.4 but still can't see much color bleeding. For specular though, the reflections in the sphere come from the bounces in that path. Took 31 min to render in a single core.

Jesús Henríquez