EC526 Project: Raytracing

Jordan Nichols

What is Raytracing?

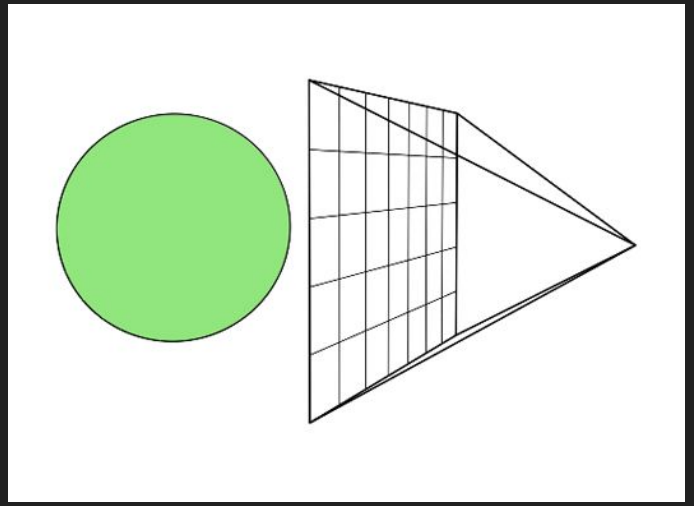
Technique which models light transport to render a scene

Pros:

- High-quality images/scenes

Cons:

- Computationally Expensive



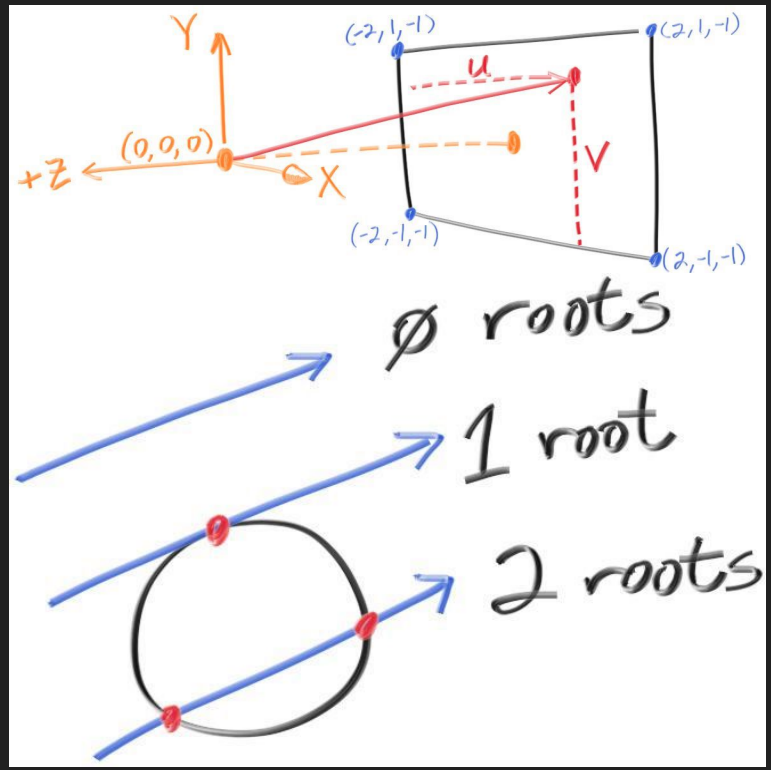
Algorithm: Setup and Simple Intersections

Definitions:

- Camera position (0,0,0)
- Viewport position
  - Rays start at camera and pass through viewport

Intersections:

- Use spheres
  - Simple to calculate intersections



Algorithm: Simple Red Sphere

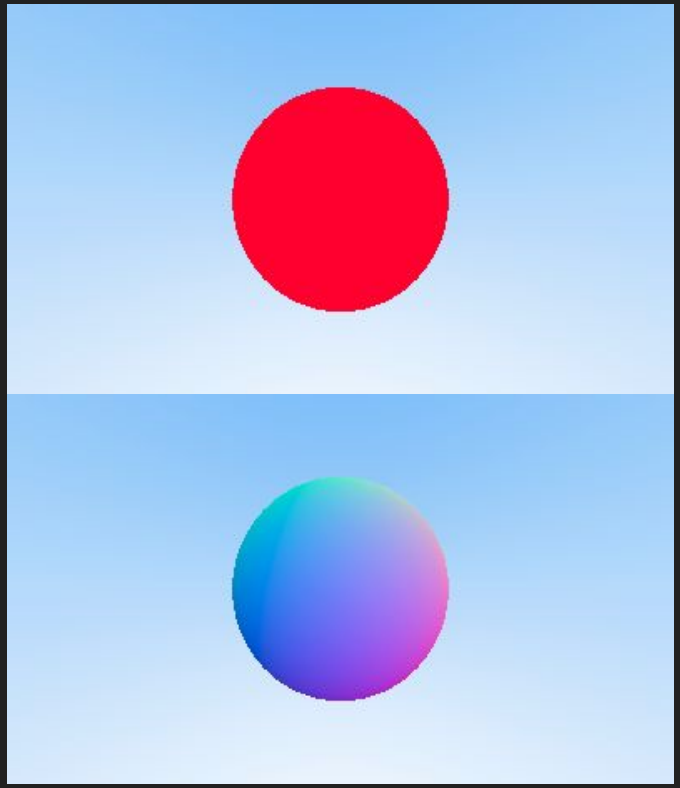
- If no intersection is detected, set to blue gradient based on y-axis

Top:

- Set pixel colors to red if ray intersects

Bottom:

- Set pixel colors based on surface normal if ray intersects



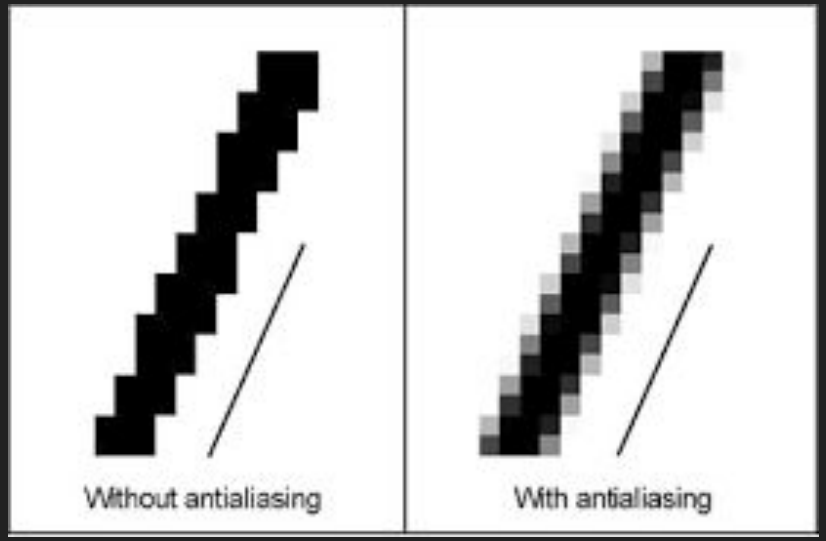
Algorithm: Smooth Edges

Problem:

- One ray per pixel means that no blending will occur and the edges of objects will be jagged

Solution:

- Take multiple samples per-pixel



Algorithm: Diffuse Materials

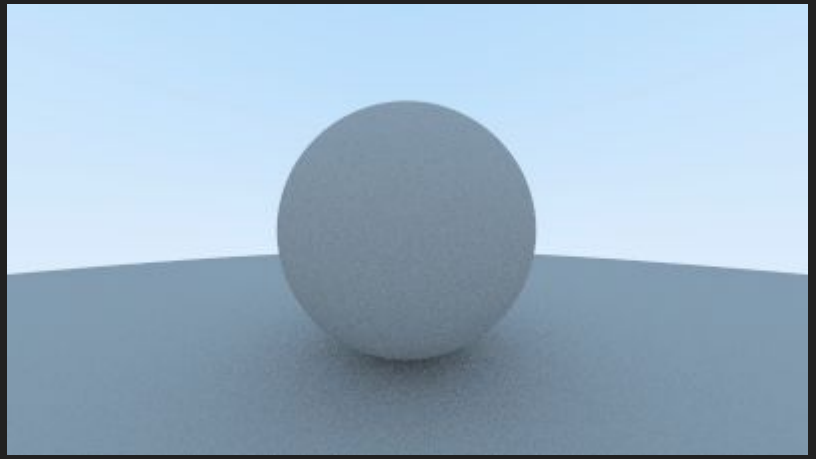
- Light rays scatter randomly off of surface
- Rays are modulated with intrinsic color of object
- Light could also be absorbed

Problem:

- Rays could reflect and scatter indefinitely

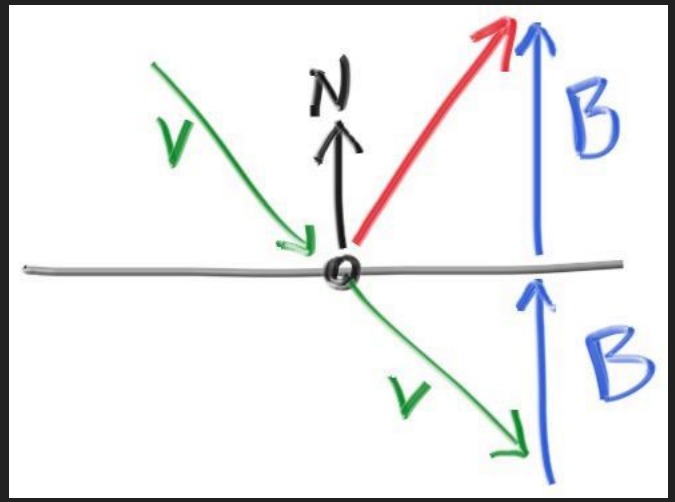
Solution:

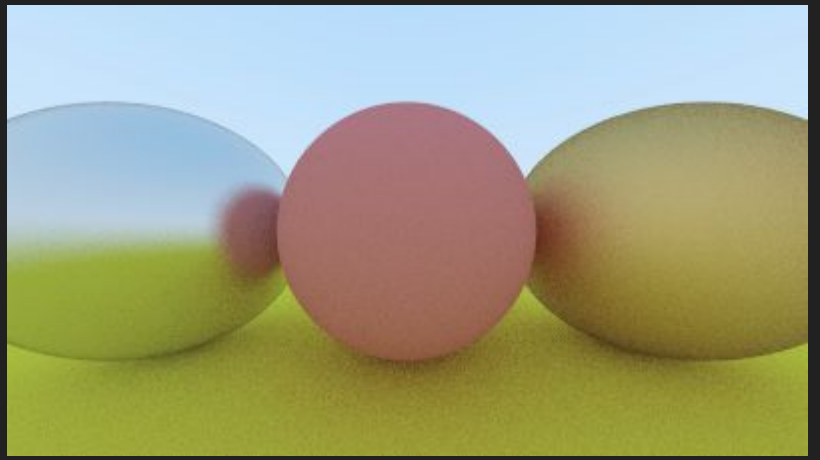
- Add “max depth” parameter



Algorithm: Metal Materials

- Light rays scatter predictably off the surface
- Can add reflection randomization as a parameter to control “fuzziness”





Algorithm: Dielectric Materials

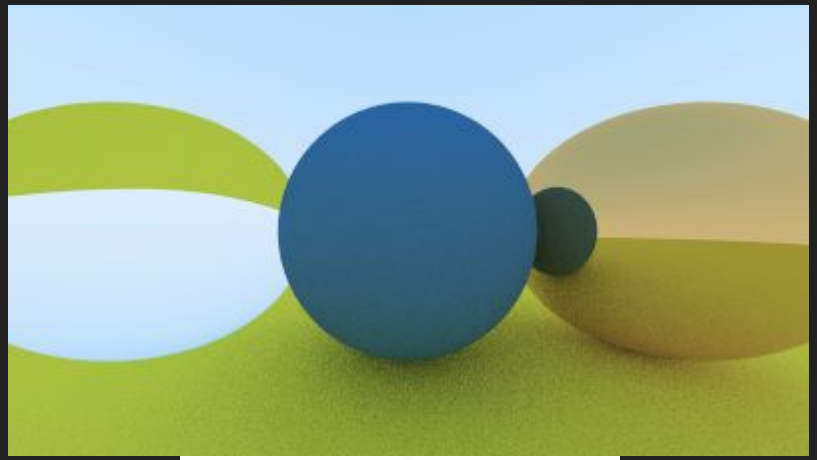
Problem:

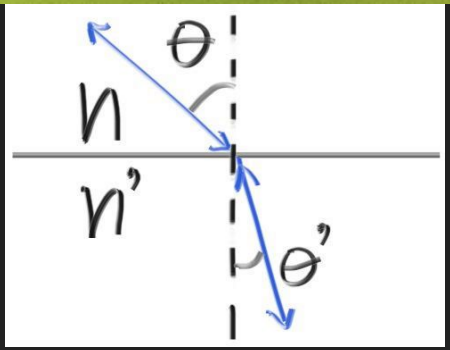
- Ray must be split into reflected ray and refracted ray

Solution:

- Randomly choose between reflection or refraction, only generating one per interaction
- Refraction calculated using Snell's Law

$$\eta \cdot \sin\theta = \eta' \cdot \sin\theta'$$





Algorithm: Pipeline

1. Loop through each pixel in the image
2. For each pixel, generate a specified number of rays (each being offset by a random value)
3. Trace each ray’s path from the camera position into the scene, determining the color of the pixel by calculating which objects the ray intersects
4. Average the colors of the rays for each pixel and write the value to an output file

Code

```
void generate_image_scalar(hittable_list world, camera cam, float aspect_ratio, int image_width, int image_height, int samples_per_pixel, int max_depth) {
        ofstream outfile_img;
        outfile_img.open("test_scal.ppm");
        outfile_img << "P3\n" << image_width << ' ' << image_height << "\n255\n";
        for (int j = image_height-1; j >= 0; --j) {
            for (int i = 0; i < image_width; ++i) {
                color pixel_color(0,0,0);
                for (int s = 0; s < samples_per_pixel; ++s) {
                    auto u = (i + random_double()) / (image_width-1);
                    auto v = (j + random_double()) / (image_height-1);
                    ray r = cam.get_ray(u, v);
                    pixel_color += ray_color(r, world, max_depth);
                }
                write_color(outfile_img, pixel_color, samples_per_pixel);
            }
        }
        outfile_img.close();

}
```

Optimization: OpenMP