Ray Tracing In One Weekend - GPU

This project is based on the InOneWeekend part (code under src/InOneWeekend) of this repo: https://github.com/RayTracing/raytracing.github.io.git
The above repo implements a CPU-based raytracer, where each pixel is sampled sequentially. In this project, I implemented identical functionalities on GPU via OpenGL and GLSL and achieved a **150x** speed-up (600s/frame -> 4s/frame) compared to the original CPU-based implementation.

Environment

- OS: Windows 11
- CPU: Intel Core i5-1340P
- GPU: Intel Iris Xe Graphics

Key differences

The main rendering loop in the original implementation is like below, where only 1 pixel is sampled at a time.

```
// Part of src/InOneWeekend/camera.h
void render(const hittable& world) {
    initialize();

    std::cout << "P3\n" << image_width << ' ' << image_height << "\n255\n";

    for (int j = 0; j < image_height; j++) {
        std::clog << "\rScanlines remaining: " << (image_height - j) << ' '
<< std::flush;
        for (int i = 0; i < image_width; i++) {
            color pixel_color(0,0,0);
            for (int sample = 0; sample < samples_per_pixel; sample++) {
                ray r = get_ray(i, j);
                pixel_color += ray_color(r, max_depth, world);
            }
            write_color(std::cout, pixel_samples_scale * pixel_color);
        }
    }

    std::clog << "\rDone.                 \n";
}
```

This is obviously inefficient and can be parellelized. By offloading all the calculation of rays (refraction, reflection, scattering, etc.) to compute shaders and dispatching them to the GPU

cores, the sampling can be done in parallel, leading to the speed-up mentioned above. Below is the snippet corresponding to the above rendering loop.

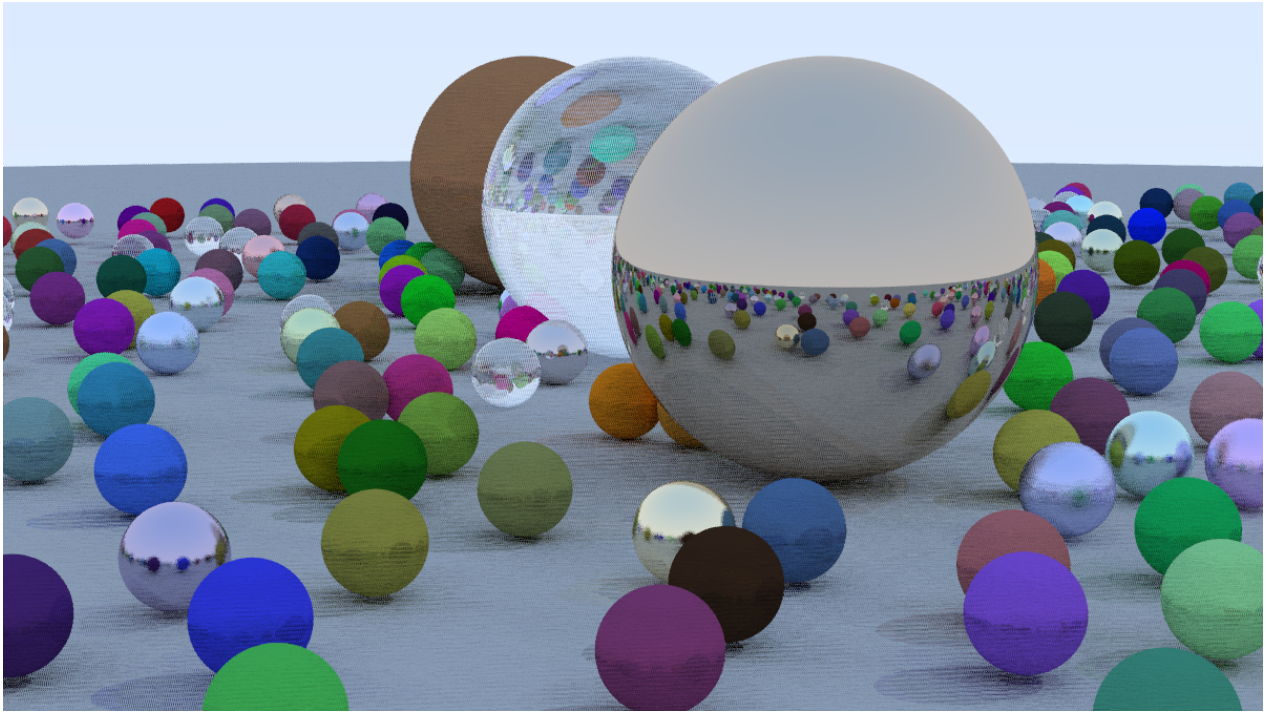
```
glUseProgram(computeProgram);
glUniform1i(glGetUniformLocation(computeProgram, "samples_per_pixel"), 10);
glUniform2f(glGetUniformLocation(computeProgram, "resolution"),
(float)image_width, (float)image_height);
glUniform1i(glGetUniformLocation(computeProgram, "num_spheres"),
num_spheres);
glUniform1f(glGetUniformLocation(computeProgram, "time"),
(float)glfwGetTime());
glDispatchCompute((image_width + 7) / 8, (image_height + 7) / 8, 1);
glMemoryBarrier(GL_SHADER_IMAGE_ACCESS_BARRIER_BIT);
```

Controlled variables

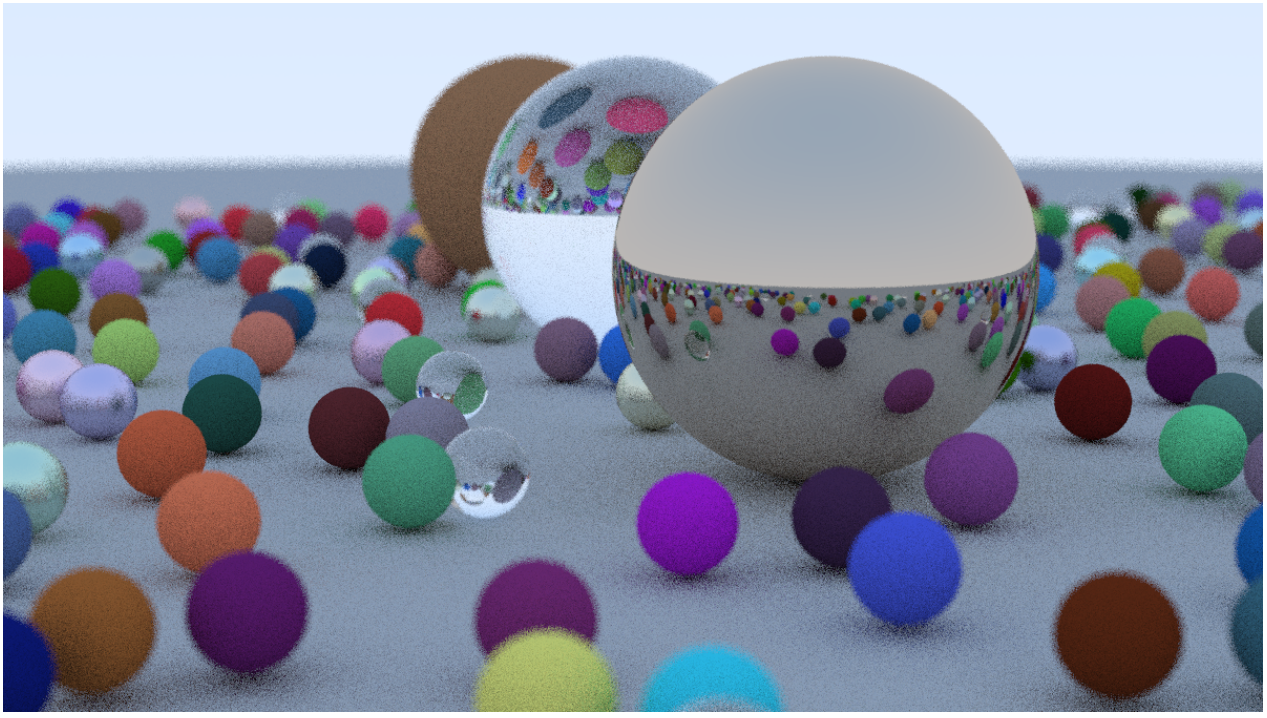
- Samples per pixel: 10
- Maximum ray hit calculations: 20
- Image resolution: 1200x675

Results

- My Implementation



- Original



References

https://raytracing.github.io/books/RayTracingInOneWeekend.html
https://learnopengl.com/Introduction