TRACE_IT

Video Rendering

Based on *Ray Tracing in One Weekend*, up to Section 12

Overview

So long for images, how about rendering videos?!, Sounds a lot cooler, right? Well, this assignment explores the creation of a ray-traced **9–10 second video**, composed of 250 individually rendered images at 25 frames per second (FPS). The objective is to simulate a *cinematic camera orbit* around a surreal, reflective and refractive 3D scene.

The task builds upon the techniques and concepts learned in Sections 1–12 of *Ray Tracing in One Weekend*, including:

- Ray-object intersections (spheres)
- Diffuse, metallic, and dielectric materials
- Recursive ray tracing for lighting and reflections
- Camera positioning and field-of-view control
- Antialiasing using stochastic sampling

Core Idea

You are free to be creative and wild ny directing your own scene, but make sure the scene is changing and dynamic, not 10 seconds of still frame. You can also use the below guidelines in which the camera:

- Orbits 360° around a central glass sphere
- Captures dynamic color and lighting changes in the surrounding objects
- Optionally incorporates effects like chromatic dispersion, object motion, and fog

Camera Animation

For frame i ($0 \leq i < 250$):

- Let $\theta = \frac{2\pi i}{250}$ (full circle)
- Set camera position: `lookfrom = (5*cos(`θ`), 2, 5*sin(`θ`))`
- Target point: `lookat = (0, 1, 0)`

- Use standard up vector: `(0, 1, 0)`
- Maintain a fixed field-of-view and aspect ratio

This creates the effect of a smooth orbiting camera flying around the scene.

Scene Design

- Large ground Lambertian sphere (e.g., y = -1000)
- Central dielectric (glass) sphere
- Several metallic and diffuse spheres surrounding the center
- Optional: time-dependent transformations (e.g., changing color, materials, positions)

Rendering Pipeline

1. For each frame index i, update the camera and optionally the scene
2. Render using 100+ rays per pixel for antialiasing
3. Apply gamma correction ($\gamma = 2.0$)
4. Save output as `frame000.ppm`, `frame001.ppm`, ..., `frame249.ppm`

Combining Frames into a Video

FFmpeg is a powerful open-source command-line tool for processing multimedia data. It can record, convert, and stream audio and video in nearly any format. In the context of this task, FFmpeg is used to combine a sequence of rendered images (e.g., .ppm frames) into a high-quality video file (e.g., .mp4), making it ideal for ray tracing animations. It is available on all platforms, see how to install for your system(mac, linux, win). Use `ffmpeg` to compile frames into an MP4 video:

```
ffmpeg -framerate 25 -i frame%03d.ppm -c:v libx264 -pix_fmt yuv420p output.mp4
```

Cool Optional enhancements:

- Add audio: `ffmpeg -i output.mp4 -i music.mp3 -shortest final.mp4`
- Apply fade-in/out or blur in post-production

Deliverables

Consider this assignment very important and Challenging so start early. I expect you to actually write some code. You have the book for reference which has already all the code, but please **DO NOT USE AI GENERATED CODE**, some people think I will not check or know, trust me a glimse is enough to tell if the code is AI generated. You can use AI to resolve your doubts but not to generate code for this. I expect.

- Fully commented Codebase with camera and scene animation logic

- All PPM frames
- Final MP4 video (9–10 seconds, 25 FPS)
- (Important) Brief write-up (1 paragraph on design choices, your experience).
- Organize all these files into folders and create a Zip file and name it in the format, rollno_week4.zip.
- This zip will be your submission.

How to submit?

Will be shared later.