Ray Tracing Project

Overview

In this project, you will develop a basic ray tracer by following the tasks in order. Implementing a ray tracer is a critical step in developing a feeling for how fundamental computer graphics techniques are implemented.

There will be required, optional, and extra tasks. You must do all the required tasks, else, your project will not be graded. Do all the optional tasks to get the full points. With the extra tasks, you can get even more than 100Pt, which will be reflected in your final grade.

- 70Pt, Required Tasks: 1, 4, 6, 8
- 30Pt, Optional Tasks: 2, 5, 7
- 5Pt, Extra Tasks: 3

You will find references to multiple resources for each task. However, you may need to read more of the resource material to understand the referenced chapter.

You should put each task's outcome in your report. Make it clear which part of your report belongs to which task. Extra commentary is always welcomed. The renders you will generate will take quite a space, so unless said otherwise they should be in 640×480 pixels. *Hint:* To put ppm images into your report, you can utilize an online file converter.

What to Submit

- The whole project in zip format, in case you are requested to compile and run your program again. Don't forget to delete the generated binaries and user-specific files, keep it small sized.
- Executable ray tracing program and minimal documentation (does it take any parameters, where is the output, etc.).
- The report in pdf format.
- Renders you generated in their original ppm format and their transformed png or jpeg format. Name your images as "<*task name*> <*enumeration*>", e.g. "Task 1 Basic Scene 0.png".

Resources

[1] P. Shirley, "Ray Tracing in One Weekend," [Online]. Available: https://raytracing.github.io/books/RayTracingInOneWeekend.html. [Accessed October 2020].

[2] "Supersampling," [Online]. Available: https://en.wikipedia.org/wiki/Supersampling. [Accessed October 2020].

[3] "Scratch a Pixel," [Online]. Available: https://www.scratchapixel.com/. [Accessed October 2020].

[4] P. Shirley, "Ray Tracing The Next Week," [Online]. Available: https://raytracing.github.io/books/RayTracingTheNextWeek.html. [Accessed October 2020].

Task 1 [Required]: Basic Scene (15Pt)

Instructions

For this task, you are expected to render multiple spheres. To start, you should follow week 3's lab session and create a scene with multiple spheres and a camera. Add a background other than the one shown in the lab. The background will act as a sky, or in shading terminology ambient, and will be the only light source when you will do the shading, so try to pick bright colors. You can see Ray Tracing in One Weekend section 4.2 [1] for a bright sky background implementation. For shading, you will need to know the normal vectors of the surfaces. Implement a way to find the normal of a ray-sphere intersection, see Ray Tracing in One Weekend section 6.1 [1].

Note: Even the section 6.1 is sufficient to compute the normals of a sphere, you are advised to read sections 6.3, 6.4, 6.5, and 6.6. They will guide you to create a scalable architecture for your renderer.

Outcome

Compose a scene with at least 5 spheres and write down each sphere's position and radius. Create 2 renders of that scene from different camera positions. Write down the camera's eye position and look-at position for each render. Use the normal vectors you computed as color values in your renders, as done in section 6.1 [1].

Task 2 [Optional]: Anti-Aliasing (10Pt)

Instructions

If you look at your renders closely you might notice that they have an aliasing issue: the borders of the spheres appear jagged. This is because currently just a single ray is traced for one pixel., and the ray is either hit a sphere or miss. However, one pixel can be in between as well. To solve this issue you can generate many more rays per pixel and combine their colors. How you generate those rays are up to you since there are many ways to do it, see Supersampling section Supersampling patterns [2]. Try at least 2 of them.

Outcome

Without changing the scene, create a render with each supersampling pattern you have tried, and a render without supersampling. Crop the same area of the images where the anti-aliasing is easily noticeable. Put them in your report side by side with labels. Discuss which one you would pick as the best. Address what anti-aliasing contributed to the render. See Ray Tracing in One Weekend Image 6 [1] for a similar comparison image. (you don't have to crop the ppm images)

Task 3 [Extra]: More Shapes (5Pt)

Instructions

Spheres are good for a start. However, they look the same from every direction which makes them a bit dull. You can implement at least 2 other implicit shapes and their normal calculations to get the extra points. You can find the implicit representations of plane and disk on Scratch a Pixel article A Minimal Ray-Tracer: Rendering Simple Shapes (Sphere, Cube, Disk, Plane, etc.) [3]. You are of course free to find much more interesting shapes online.

Outcome

Create a render for each new shape you added. Try to compose scenes that demonstrate their geometries. Don't forget to add the references to where you found these implicit formulas.

Task 4 [Required]: Diffuse and Metal Materials (25Pt)

Instructions

Now it's time to put in use of those normals and do some shading. You will implement shading methods for diffuse and metal materials. Diffuse materials reflect the light with total randomness, they are chaotic. Metals on the other hand reflect the light perfectly, they are orderly. Though this doesn't mean that all the metal materials will look like perfect mirrors, you can add an amount of randomness to the reflection vector and achieve a degree of blurriness. You will implement the Lambertian Reflectance model to simulate the diffuse materials, see Ray Tracer in One Weekend section 8.5 [1]. For the metal surfaces, you will implement perfect reflectance and add a parameterized randomness to it, see sections 9.4, and 9.6 [1].

With the introduction of shading and bouncing the rays, some artifacts appear. One of them is the incorrect gamma values of the colors. You should fix it by following section 8.3 [1]. The second artifact is called shadow acne. Due to the finite precision of floating-point numbers, some of the intersections return a point that is not on the surface but slightly inside it. In such cases, the rays fail to reflect the surface but stuck inside the shapes, creating incorrect (mostly dark) spots on the render. To solve it, you should cast the reflection rays with a small margin to the intersection point, see section 8.4 [1].

Note: The implementations on sections 8.1, and 8.6 [1] mentions alternative models for diffuse materials, reading them may give you a broader idea about how these models are developed.

Note: Again, similar to the shapes' implementation, there are some extra steps you should do to achieve a good architecture. See Ray Tracer in One Weekend sections 9.1, 9.2, and 9.3 [1].

Outcome

Create a render with at least 4 shapes and use both the diffuse and metal materials. Write down the material properties you have used. From now on, due to the randomness introduces in the materials, the colors returned from two identical rays might be very different. You should pick a large sample size to achieve smooth renders. Know that this might take a long time, so plan ahead.

Task 5 [Optional]: Refraction (10Pt)

Instructions

The diffuse and materials are good but they are opaque. There are many other materials such as glass, water, ceramic, and honey that are not opaque or partially opaque. These materials are called dielectrics and they do not reflect the light but refract it, meaning that light passes through them. To simulate the refraction behavior of light you can apply Snell's Law, see Ray Tracer in One Weekend section 10.1, 10.2, 10.3, and 10.4 [1].

Outcome

Create at least 1 render that demonstrates the new material you have implemented. Explore how the index of refraction affects the refraction. Write down the material properties you have used.

Task 6 [Required]: Lights (15Pt)

Instructions

Until this point, there were no lights in the scene besides the background (or ambient). For this task, you will create a material that emits light instead of reflecting or refracting it, see Ray Tracing The Next Week section 7.1, and 7.4 [4]. The light sources can be simulated conceptually as well. We saw such lights as point light, spot light, and sun light, in week 1's Blender demonstration. However, giving surfaces an emissive material suits better when you try to render physically accurate scenes because in reality all light sources also have a physical body.

Outcome

Create 2 renders. Both should include at least 2 shapes with emissive materials. At least 1 of them should have no background (ambient) light, see section 7.2 [4]. *Tip:* Try to give your lights a color other than white.

Task 7 [Optional]: Let's get creative! (10Pt)

Instructions

You can get as creative as you want with this task. Use the features you added into your renderer and create a visual that you like. Just make sure that it shows the effort you put in.

Outcome

At least 1 render with a minimum of 1920×1080 pixels. Commentary is appreciated.

Task 8 [Required]: Questions (15Pt)

Instructions

Answer the following questions:

- The approximate time complexity of ray tracing on models with triangles.
- What is the difference between preprocessing and computing the image? Why?
- What are the critical parameters for the ray tracer algorithm's performance?
- Imagine you are a systems engineer at Pixar. There is a new super-resolution in 6000×6000 pixels and you have to estimate the maximum rendering time per frame. Assume that the scenes are static, so you are not going to spend any CPU on animation, scene hierarchy, etc. The average scene has 500 objects with a total of 5.000.000 triangles to check intersection with.

Outcome

Write down the answers in your report.