CAP 5705 Computer Graphics
Dr. Corey Toler-Franklin

Program #2 Ray Tracing
DUE: October 2nd, 11:59 pm

Overview | Details | Resources | Getting Help | Submitting

Overview

The ray tracing algorithm generates photorealistic images from 3-D geometry in a scene. Ray tracers simulate the behavior of light as photons travel from a light source, collide with and bounce off of objects in the scene, and eventually reach the observer. This process can be computationally intensive depending on the scene complexity and number of light sources.

You will implement a ray tracer that generates images of simple scenes composed of spheres and triangles. Your work will be evaluated based on the following criteria (1) source code completion and correctness 30% (2) ray intersection 20% (3) shading algorithms 20% (4) special effects 20% (5) one page report 10%.

Details

Getting Started

You are free to alter the provided coding framework to accomplish your task. The framework is simply a guide. Extract the file proj2_raytracer.zip to your local folder. The source code framework is in the *src* folder. Datasets (meshes and textures) are found in the *data* folder. Other helpful materials are in the *docs* folder. You may add folders and files to your project zip file but keep the same directory structure for the framework you were given.

To build the proj2_raytracer application on linux, type *make* in the *src/mainsrc* folder. This also generates a visual studio c++ solution file. Use any platform but make sure your code compiles again using the make file on linux before you submit. If you need help selecting a debugging interface, see the TA. Included in the package is libst, an open source openGL wrapper, and tinyojloader (by Syoyo Fujita), an OBJ file object loader. The program is executed from the main function in the file mainsrc/mainsrc_proj2.cpp. Examine the classes *RayTracer*, *Ray*, *Intersection*, *Surface*, *Triangle*, *Sphere* and *Shader*. You will use function declarations and code comments as guides to complete these classes which have been partially implemented for you. **Search the project files for *TO DO: Proj2 raytracer*. In the following sections, you will insert or alter code at these locations to complete the assignment.**

Intersections

Surfaces Complete functions that implement ray surface intersections. You are given a base class, *Surface* and sub-classes *Sphere* and *Triangle*. Implement the intersection functions in the Sphere and Triangle sub-classes. The input is a ray and the outputs are the intersection information and a true flag (if found) or a false flag (not found).

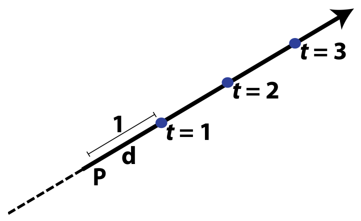

Figure 1: Defining a Ray

As discussed in class, your implementation should cast a ray (Figure 1) from the eye (camera) position and test for intersections along the ray (at time intervals **t**) with surfaces. The direction of this path is the reverse of what happens in reality. This approach makes the problem tractable by insuring that only photons that actually reach the observer are considered. The intersection function returns true if an intersection is found; otherwise false. Update the Intersection class with information about the computed intersection.

Scenes Next, complete code that computes ray to surface intersections for every object in the scene. The code will visit each surface in the scene and call the *Surface* intersection function on each *Surface*. Only store the closest intersection points (Remember our discussion about shadowing and occlusion). Figure 2 illustrates the ray path and intersection.

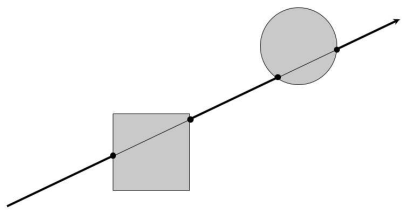

Figure 2: Ray to Surface Intersections

Shading Now implement the *Lambertian* function in the *Shader* class. The Lambertian Reflectance Model, covered in class, describes the relationship between the observed intensity values in an image and the normals (surface orientation information) of the objects in the scene. Figure 3 illustrates that this relationship is proportional to $\cos(\theta)$ where θ is the angle between the incident light direction l and the surface normal $\hat{n}$. The model assumes a perfectly diffuse surface. Once you have implemented the Lambertian function, implement the *Phong* function. Refer to the textbook and lecture slides for the formulation.

Ray Trace You now have code for a very basic ray tracer. Use it to implement the ray tracing algorithm in the *Run* function of the *Ray Tracer* class. The three stages are: (1) cast a ray from the camera into the scene, (2) compute intersections with surfaces, (3) shade objects in the view and store the results in an image. You will be able to add more complex functionality later after we cover advanced topics.

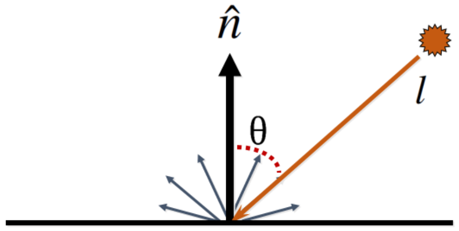

Figure 3: Lambertian Reflectance Model

Special Effects

Word has gotten around that you are a computer graphics wiz. You have been hired by BlueSky Studios to create a special effect for a new commercial. Add additional appearance properties to your Surfaces to render a special effect of your choosing. Consider adding Fresnel shading, mirror reflection or transparency. Use the textbook and lectures for ideas. Your Intersection class contains a pointer to the Surface found at the point of the intersection for access to surface properties in the Shader class.

Report Write a one page document that includes the following:

- Three ray traced results: Lambertian Shading, Phong Shading and a Special Effect
- Describe your approach for creating the special effect.
- List any difficulties you had with the assignment.

- Add any special instructions I need to run your code.
- Explain any challenges (e.g. Why your code does not run. Bugs?)

Resources

The openGL Programming Guide, available online https://www.opengl.org/sdk/ is a good reference. Appendix A also provides information about GLUT and GLEW. Chapter 12 of the textbook, Fundamentals in Computer Graphics, covers meshes. Information about the *.obj* file format is in the *docs* folder.

If you are not in the CISE department, and would like computer lab access, you can register for a CISE account at https://www.cise.ufl.edu/help/account. The source packet proj1_mesh.zip will run on machines in the computer labs on the first floor of the CSE building. Labs are open 24 hours (see https://www.cise.ufl.edu/help/access). **Remember to back-up your work regularly!!!** Use version control to store your work. DO NOT PUBLISIZE SOLUTIONS.

Getting Help

Source Code Please do not re-invent the wheel. Use the source code framework (and comments) and review the notes in the docs folder.

Discussion Group Post questions to Canvas (everyone benefits in this case), or send me an email at ctoler@cise.ufl.edu. I will check both daily.

Office Hours Stop by my office, CSE 332 (or lab CSE 319) during office hours MWF 10:40 to 11:30am or see the TA in room 339 TH 3:00 to 5:00pm.

Collaborating Work independently. Remember to always credit outside sources you use in your code. University policies on academic integrity must be followed.

Submitting

Upload your proj1_mesh.zip fie to Canvas. It should include:

- Source code.
- Make file.
- One page report.
- Three ray traced images: Lambertian Shading, Phong Shading, Special Effect.