Assignment 2 – Ray Tracing (Part 1 – Rays, Lights & Spheres)
Due: October 11th

Overview:

In this assignment, you will create a simple ray-tracing application capable of rendering scenes consisting of spheres lit by point light sources. Scenes will be described in a simple, easy to parse text specification file. This is part 1 of a two-part assignment. In the second part you will extend this raytracer to incorporate triangle meshes, more complex lighting, and other extensions for more realistic images.

Getting Started:

You already have the `stb_image_write.h` library from HW1. You can use this to create BMP outputs. You may also wish to use SDL or GLFW to create an interactive UI for displaying and modifying your scenes. All other code you should write from scratch.

A simple scene format has been posted online [https://z.umn.edu/2zpp], this contains information on the camera, all objects, lights and materials in the scene, and output formats. You may need to extend it to implement new features, but be sure you can also support existing files in this format. We have uploaded some scenes consisting of only spheres to get you started [https://z.umn.edu/2zpq]. *NB*: My rendering of these scenes contains some features such as reflection and refraction, which you do not need to implement until Part 2.

Submission Instructions:

Create a sample webpage with:

-Your source code

-Output images

-You must show at least one of the sample scenes along with several new scenes of your own designed to show off each feature of your raytracer.

-Brief description of your implementation, any issues you saw, and a list of any extra credit tasks you attempted

-Submission to Art Contest (optional)

Assignment Requirements:

As this is a two-part assignment, Part 1 will be graded out of 50 pts. The number is front is how many points the feature is worth. Partial credit will be given to features that "mostly" work. The required minimum feature set is underlined.

Scene Setup:

(5) Camera placement, film resolution, aspect ratio

(5) User specified background colors

(5) UI + OpenGL output

(5) BMP or PNG output

Primitives:
(5) Spheres
(5) Difference/Intersection of spheres (Constructive Solid Geometry)

Lighting:
(5) Ambient lights
(5) Point light sources
(5) Shadows
(5) Multiple light sources
(5) Directional light sources
(5) Spot light sources

Sampling:
(5) Basic Sampling
(5) Jittered Supersampling
(5) Adaptive Supersampling
(5) Motion Blur
(5) Depth of Field

Materials:
(5) Color & Specularity (Phong Lighting Model)
(5) Refraction
(5) Reflection

Hints:

-Test your program incrementally

-Try hard to test incrementally

-Honestly, incremental testing will really help!

-By testing incrementally I mean, you should get some very small thing working first then move on to more complex features. First generate a black image of the correct dimensions. Then create a scene with just a sphere. Have your raytracer return white whenever it hits the sphere. You should see a white sphere on a black background. Now make sure you feel confident in your ray/sphere intersection code. Move the sphere around, make sure the right results happen. When this works, mess around with the image aspect ratio (make it very narrow), make sure everything works as expected. Then instead of coloring the sphere white use the real material color. Then add support for point lights, then phong shading. Only move on to a new feature when you're sure everything else works so far.

-If you don't do this, you'll end up with a big pile of broken code and neither I nor the TAs will be able to help you.

-Test incrementally

-Please, please test as you go along!

-See the posted HW2 strategy guide