CS430 Computer Graphics

Project 3 – Illumination

In the previous project you will wrote code to raycast mathematical primitives based on a scene input file into a pixel buffer. In this project you will color objects based on the shading model we discussed in class.

Your program should be resistant to errors and should not segfault or produce undefined behavior. If an error occurs, it should print a message to stderr with "Error:" prefixed to a descriptive error message before returning a non-zero error code. I have a test suite designed to test the robustness of your program.

Your program (raycast) should have this usage pattern:

```
raycast width height input.json output.ppm
```

The JSON data file should support all the primitives from project 2 and should implement a new light primitive. Examples of the fields for lights follow:

```
[
{"type": "camera",
 "width": 2.0,
 "height": 2.0},
{"type": "sphere",
 "radius": 2.0,
 "diffuse_color": [1, 0, 0],
 "specular_color": [1, 1, 1],
 "position": [0, 1, 5] },
{"type": "plane",
 "normal": [0, 1, 0],
 "diffuse_color": [0, 1, 0],
 "position": [0, -1, 0] },
{"type": "light",
 "color": [2, 2, 2],
 "theta": 0,
 "radial-a2": 0.125,
 "radial-a1": 0.125,
 "radial-a0": 0.125,
 "position": [1, 3, 1]}
]
```

Specifically, these properties should be supported for lights:

`position`	The location of the light
`color`	The color of the light (vector)
`radial-a0`	The lowest order term in the radial attenuation function (lights only)
`radial-a1`	The middle order term in the radial attenuation function (lights only)
`radial-a2`	The highest order term in the radial attenuation function (lights only)
`theta`	The angle of the spotlight cone (spot lights only) in degrees; If theta = 0 or is not present, the light is a point light; Note that the C trig functions assume radians so you may need to do a conversion.
`angular-a0`	The exponent in the angular attenuation function (spot lights only)
`direction`	The direction vector of the spot light (spot lights only) If direction is not present, the light is a point light

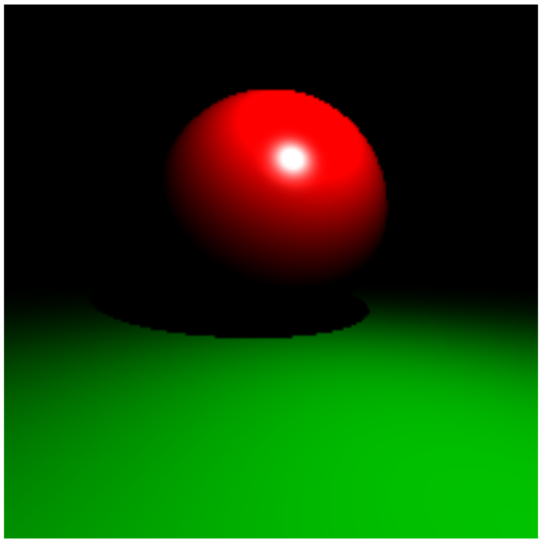
For objects, the properties from the last assignment should be supported in addition to:

`diffuse_color`	The diffuse color of the object (vector)
`specular_color`	The specular color of the object (vector)

You may optionally include an object property, **`ns`**, for shininess. If the property is not present please set the value to **20** (you may simply hard code the value in the specular equation).

I will do some basic JSON error checking but you may assume that the properties in your scene are consistently set. I will not, for example, set theta to zero and then set angular-a0 to some value. Nor would I set radial-a0 on a sphere.

When rendered the previous sample should look something like this:



Technical Objectives

Technical objectives describe the organizational or code-related features that are a required part of your application and will be evaluated in the technical objective rubric for this project. In grading technical objectives, we will ask the question "How well does this project provide evidence of the objective?" For example, a single Git commit probably does **not** represent outstanding "use of git".

- Ability to read JSON scene files
- Ability to shade primitives
- Ability to represent lights
- Ability to render shadows
- Use of C programming language
- Use of consistent coding style and commenting
- Use of Git

Creative Objectives

Creative objectives mirror the technical objectives but involve subjective creative features of your project. In this project you should create a demonstration scene. For this project the creative objective has very little weight but I still encourage you to consider these objectives:

- Uses a range of colors
- Makes the correct orientation of the scene obvious (which way is up, left, right, etc.)
- Visually interesting

What do I turn in?

In BBLearn you should turn in a report (entered with the BBLearn editor) with the following format:

Project # and Title

Your Name
Your NAU User ID

Link to Github Repository

Your Github repository should contain all the code for your repository. It should also include a README.md file that describes your application, usage, and communicates any special notes to the grader. A Makefile which can be used to build your project. Your program should compile with gcc without any special libraries (libc and libm are ok).

The grading rubric will be posted in BBLearn.

Graduate Student Extension (CS599)

If you are a graduate student, you should also shade quadrics. This may require some research and creative thinking.

You should include example scenes for a cylinders, cone, and ellipsoid in your repository.