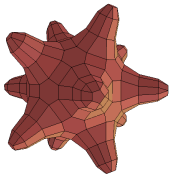

CS430 Computer Graphics

Project 4 – Recursive Raytracing

In the previous project you will wrote code to raycast and shade mathematical primitives based on a scene input file into a pixel buffer. In this project you will add recursive raytracing to provide reflection and refraction.

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 (raytrace) should have this usage pattern:

raytrace width height input.json output.ppm

The JSON data file should support all the primitives from project 3 and should support new attributes for reflectivity and refractivity:

```
[
{"type": "camera",
 "width": 2.0,
 "height": 2.0},
{"type": "sphere",
 "radius": 2.0,
 "reflectivity": 0.2,
 "refractivity": 0.3,
 "ior": 1.33,
 "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],
 "specular_color": [1, 1, 1],
 "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 new properties should be supported for objects:

`reflectivity`	The amount of reflection (0.0-1.0)
`refractivity`	The amount of refraction (0.0-1.0)
`ior`	The index of refraction of the volume

For planes assume that the volume that the index of refraction applies to is on the opposite side of the plane from the camera.

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.

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 demonstrate reflection
- Ability to demonstrate refraction
- Ability to raycast scene per Projects 2 and 3
- 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
- Demonstrates reflection and refraction

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
>
> git clone YOUR_REPO YOUR_NAU_ID

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.