ASSIGNMENT 1

João Madeiras Pereira

Due date : 19 March 2019

(Att: see below the Lab Submission section**)**

Introduction

This assignment weights 15% of the final classification. So far, we have considered only *local* models of illumination; they only account for incident light coming directly from the light sources. *Global* models include incident light that arrives from other surfaces, and lighting effects that account for global scene geometry.
Such effects include:

- Shadows
- Global Illumination like reflections of other objects, in mirrors, for example

Ray Tracing was developed as one approach to modeling the properties of global illumination. The basic idea is as follows: For each pixel:

- Cast a ray from the eye of the camera through the pixel, and find the first surface hit by the ray.
- Determine the surface radiance at the surface intersection with a combination of local and global models.
- To estimate the global component, cast rays from the surface point to possible incident directions to determine how much light comes from each direction. This leads to a recursive form for tracing paths of light backwards from the surface to the light sources.

Computational Issues

- Form rays.
- Find ray intersections with objects.
- Find closest object intersections.
- Find surface normals at object intersection.
- Evaluate reflectance models at the intersection.

Objectives

For this assignment, you will implement the naïve T. Whitted's Ray-Tracer as presented in the classroom. Your program should support:

- Parsing NFF file format **(1 point**)
- Local color component (Blinn-Phong model illumination) **(2 points**)
- Multiple source lights (**1 point**)
- Hard Shadows (**1 point**)
- Global color component by implementing the mirror reflection (**4 points**) and refraction (**4 points**)
- Ray intersections with spheres (**2 points**), infinite planes (**1 point**) and triangles (**2 points**)

- Bonus for extras (**2 points**). For instance, supporting other geometry intersection (AABBs, cylinders, cones, etc). These could be tested with NFF scenes. See SPD software. Or by allowing other scene's format as PLY.

Implementation Details

Setup

For this assignment, you can rely on the environment setup used to solve the Exercise 1.
You should use the algorithms described in the theoretical classes. For instance, you can use the Badouel's or Moller&Trumbore´s algorithms for the triangle-ray intersections.

Testing

You should test your program by using the attached NFF files regarding the *Balls* and *Mount* scenes with different level of complexity.

Lab Submission

Submit in the **Fenix system** your source code (.C and .h), and/or Makefile (if you have any) and a readme file specifying what are being submitted and how to compile and link your program.
All the files should be zipped in a file called Assignment 1.
Do not submit any executable files. We will use some sample NFF models to test your program.

The printed report with 4 pages at maximum should be delivered in the lab next week.

Late Penalty
You should submit your solution on time. Being late for one checkpoint could affect the time left for you to complete subsequent labs. All labs are due at the above specified due data, and there is a 20% penalty each day for up to 40%. After that, you get zero.

Grading Criteria
Grading of the labs will be based on the following:

- 85%: Correctness and adherence to assignment specification. Part of it will be checked on the demo provided by the Groups in the lab class regarding the checkpoint 1.
- 10%: Readability, structure of code, use of comments, adherence to lab procedures (submitting, naming conventions, etc.)
- 5%: Printed report

Don't copy labs. Discussion of lab assignments is allowed and encouraged. However, you need to complete the lab all by yourself. Labs which are too similar will be properly handled by the teaching members of the discipline.