Introduction to Computer Graphics
Assignment 1 – Planes and Cylinders

Handout date: 25.02.2020

Submission deadline: 5.03.2020, 13:00 h

Late submissions are not accepted

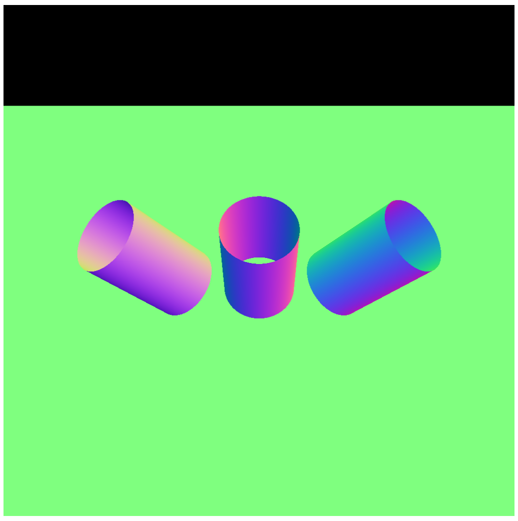

Figure 1: Expected results for `scenes/cylinders/cylinders.sce`

In this assignment, you will implement ray intersections with planes and cylinders and compute surface normals at the intersection points. The framework code provided this week is identical to last week's, except "`todo`" comments have been inserted in `Plane.cpp` and `Cylinder.cpp` to indicate where you need to add your implementations. If you already set up a GitHub repository last week to collaborate with your fellow group members, you can copy these TODO comments over to your repository (or just note where your implementation needs to go and get started).

In the `expected_results` directory, we provide the images you should expect your finished code to produce for a subset of the provided scenes. One such result is shown in Figure 1. The framework is configured to visualize the surface normals in false color (the normal XZY components in $[-1,1]$ are mapped linearly to RGB values in $[0,1]$).

Regarding the orientation (sign) of surface normals: the raytracer expects your implementation to always return a normal pointing *towards* the viewer! So for "non-closed" objects like the plane and open cylinder, where the ray may hit the surface from either side, you must choose the sign appropriately based on the ray direction. Specifically, the normal and view ray should always point in opposite directions. You can check if you got this right for the cylinder byc omparing to Figure 1.

Please submit your output on the scene `scenes/combo/combo.sce`.

Notes on Implementing Cylinder Intersections

You are asked to compute the intersection with an *open* cylinder (i.e., without end caps). The approach we recommend is to first determine where the ray intersects a version of the cylinder that extends infinitely in each direction (as if `height` $= \infty$). Then, from this list of intersection candidates, discard those that fall outside the cylinder's actual extent. Finally, choose the first remaining intersection that appears in front of the viewer.

Hint: you may find it helpful to read over `Sphere::intersect` in `Sphere.cpp`.

Theory Exercise

Derive the formulas you use to solve for the cylinder intersections and compute normals. For full credit, the derivation must be complete (ending with the formulas you implement in `Cylinder.cpp`) and possible to follow.

Grading

Each part of this assignment is weighted as follows:

- Ray-plane intersection: 20%
- Ray-cylinder intersection + normal derivations (theory exercise): 25%
- Ray-cylinder intersection implementation: 40%
- Cylinder normal implementation: 15%

What to hand in

A .zip compressed file renamed to `Exercise`N`-Group`I`.zip`, where N is the number of the current exercise sheet, and I is the number of your group. It should contain **only**:

- The files you changed (in this case, `Plane.cpp` and `Cylinder.cpp`) and the requested program output. It is your responsibility to ensure that all files that you have changed are in the zip.
- A `readme.txt` file containing a description on how you solved each exercise and the encountered problems. Indicate what fraction of the total workload each project member contributed.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.

- A `TheoryExercise.pdf` containing your cylinder-ray intersection and normal derivations.

Submit solutions to Moodle before the deadline. Late submissions receive 0 points!