Introduction to Computer Graphics
Assignment 4 – Raytracer Extensions and OpenGL "Hello World"

Handout date: 20.03.2018

Submission deadline: 29.03.2018, 13:00 h

Late submissions are not accepted

This assignment gives you an opportunity to play around with and extend your raytracer. It is also intended to ensure that you can build and run the OpenGL framework code we'll be using for the upcoming assignments. This assignment will not be graded, but Professor Pauly will select a few of the best submissions to showcase in an upcoming lecture. Please do make sure the OpenGL code is working on your computer during this assignment; in later weeks, we will not provide assistance in setting up the code.

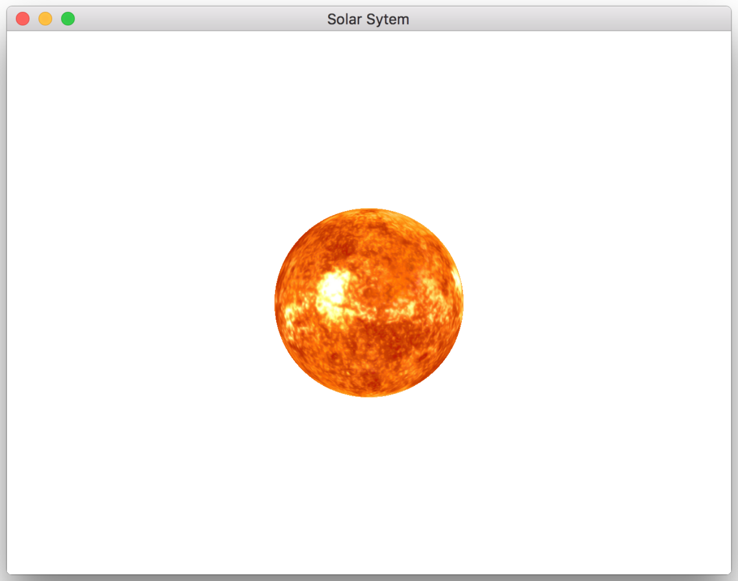
Here is a list of potential ideas for using or extending your raytracer:

- Set up an interesting scene. Create a new scene file/collection of meshes that produces a nice image. You might consider downloading some free triangle meshes from an online service (e.g., cgtrader or turbosquid) or creating your own in Blender. Note that whatever mesh you use must be converted to `OFF` format to be loaded into a scene. You can do this conversion with Blender.
- Render glass objects by implementing refraction. Refraction works similarly to reflection, though you'll need new formulas for computing the angle of the refracted ray (this can be done with Snell's law given the index of refraction of air and glass) and determining how to mix the reflected and refracted colors (this can be done with the Fresnel equations). Implementation details can be found in this article.
- Add a torus primitive and implement ray-torus intersections. You can take the same approach to deriving the intersection equation as you did for cylinder intersections. However you will end up with a quartic equation, not a quadratic one. You'll likely want to use a numerical method to solve for the smallest positive root of this equation instead of evaluating the rather complicated closed-form root expressions (see Wikipedia).
- Generate a movie. An example of how to do this is provided in `scenes/movie` of last week's assignment code. On Linux and macOS, you should be able to open a terminal in this scene directory and run `bash gen_movie.sh` (after installing `ImageMagick` and `FFmpeg`). This script assumes that you built your raytracer binary in a `build` directory inside the main project directory. The script feeds 90 different scene files into the raytracer to render each frame of animation and then stitches the output frame images into a movie.

Note that if you add new `.cpp` files for your implementation, you will need to edit `src/CMakeLists.txt` to include them.

OpenGL Framework

Please download and unpack `assignment_4.zip` and see its `README.md` for instructions on building the framework code. If all goes well, you should see the following window when you run the `SolarSystem` binary:



Grading

This assignment is ungraded.

What to hand in

A compressed .zip file renamed to `Exercise4-Group`*i*`.zip`, where *i* is your group number. It should contain:

- The source files that you added or changed in the raytracer.
- The image/movie you created.
- A screenshot of the running OpenGL assignment code.
- A `readme.txt` file describing what you added to or did with the raytracer this week. Indicate what fraction of the total workload each project member contributed. Please also tell us any difficulties you encountered in getting the OpenGL framework running.