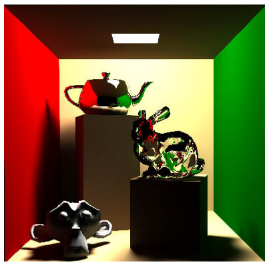
Final Project TI1806 Computer Graphics



Interactive Ray Tracing

An OpenGL-based Ray-tracing Visualization

Group size: recommended 7, minimal 5 (only admitted for good reasons), max 9.

Examination: 25th of June 2015, time slots will be determined later

Evaluation: 4 minutes sharp (!) for a presentation (Powerpoint or demo, or both)

+ 3 minutes of questions

Presentation and questions constitute 5 % of your grade, the project itself 35%.

To hand in: A bullet list of the implemented features, your presentation slides in PDF, one JPG to participate in an image competition. **The presentation and the bullet list should start with the list of all members, including their student number.**

Project Description

The final project of the course Computer Graphics will evolve around ray tracing. The goal is to develop an interactive tool to shoot a ray into a scene and visualize its trajectory – meaning its first hitpoint, (recursively) potential reflected/refracted rays, and light rays.

The minimal implementation should contain the following features:

- Ray intersection with planes, and triangles.
- Illumination of the impact point which is drawn in the resulting color
- An option to move the light sources in the scene
- A shaded display in OpenGL of the 3D scene

Although it was a bit drier than usual, the last lecture is very useful as it lists all the formulas, which you can use to perform the tests above.

In the next lecture, we will cover acceleration structures, which are an advanced feature to explore.

Of course, you can always extend your project by adding additional components. For example:

- A possibility to loop over the rays via the keyboard, triggering a ray highlighting (e.g., change of color) and a command line output of the selected rays properties.
- A scene hierarchy and ray-tracing acceleration structure
- A selection of triangles for which to modify attributes on the fly
- Perform an actual full-image ray tracing instead of a single ray
- Ray intersection with spheres
- Visualization of the acceleration structure
- Design your own 3D scene in Blender
- Create interesting test cases
- …

Code

The starting point of your project is the attached code sample, which opens a 3D model including its normals, and displays it in OpenGL (the light is following the camera).

Additionally, upon pressing on "space", a single ray is produced underneath the mouse position. This ray is stored and then displayed. When you move the camera, the ray stays visible. Currently, this ray does not interact with the geometry – which is for you to implement.

Talking of implementation, you really only need to take a look at four files:

raytracing.h/cpp: this is the file you mainly work in
mesh.h (and Vertex.h): the mesh structure, against which you will trace rays
(**Vec3D.h**: vectors to describe vertex positions --- but you already know this one)

There are many comments in the code, please read through them carefully.

The compilation is the same as for your last exercise. In fact, you can take the last solution file, open it in Visual Studio, remove all files from the solution and insert all new files instead.

ATTENTION: YOU NEED TO CHANGE THE FILE PATH in the "init" function of your raytracing.cpp

Organization

You can subscribe to groups on Blackboard right now. To find group members, please don't hesitate to use the discussion forum. If you cannot find group members by the end of the week, please send an email to: L.Scandolo@tudelft.nl on **Monday morning** with the **subject "TI1806 Looking for a group"**. We will then find a solution for you.

By next week, once the groups are established, you will receive the name of a TA to contact in case of urgent questions. Nonetheless, as you are numerous and have many members in a group (and colleagues), you should first try to solve the issue internally. Please notice that the TAs will not be able to write code for you… otherwise, where would be the fun? ☺

Good luck!