Course: “Computer Graphics”, ECS 175, Fall Quarter 2015
Instructor: Bernd Hamann

Project 5: **“A SIMPLE RAY TRACER”**

Date due: Thursday, December 3, 2015

The fifth project requires the implementation of the **ray tracing** algorithm discussed in class. Write a program to render a scene in 3D space containing planes and other implicitly defined surfaces of degree 2 (*e.g.*, spheres and ellipsoids), and triangulated surfaces. The **input parameters** for the program are the **from point**, the **at point**, the **up vector**, and the **viewing angle** α. The **position of the light source(s)** and the **resolution** of the final image ($N \times N$ pixels) will be specified by the user as well.

You must implement the *generalized* Phong illumination model considering direct and global illumination effects, given by the formula

$$\begin{aligned} I &= I_{\text{direct}} + I_{\text{global}} \\ &= k_a I_a + \frac{I_{\text{light}}}{K + \text{distance}(\mathbf{f}, \mathbf{p})} \left(k_d(\vec{\mathbf{l}} \cdot \vec{\mathbf{n}}) + k_s(\vec{\mathbf{r}} \cdot \vec{\mathbf{v}})^n \right) + k_r I_r + k_t I_t, \end{aligned}$$

where the Phong illumination formula now incorporates the **global illumination** term I_{global}. The values I_r and I_t are vector-valued (red, green, and blue components). They are obtained by applying the Phong illumination model recursively. All **parameters in this equation are input** for the ray tracer. The user can specify the color properties of each object in the scene. The global illumination term must be **computed recursively** as discussed in class. When computing the color/intensity for a particular pixel, stop the recursion when a user-specified **maximum number of recursion levels** is reached or when a reflected/refracted **ray hits one of the faces of the bounding box** surrounding the given scene.

For this project, you need to consider **intersections** between rays (=lines defined in parametric form) and implicit surfaces and between rays and triangles approximating surfaces. Use the intersection algorithms discussed in class. In order to allow **transparent objects**, you also need to implement the procedure for computing refracted rays. This requires the specification of **refraction coefficients** η for all objects/media in the scene. The user must be able to change these. **Shadow feelers** must be used at each point encountered in the scene to determine whether it receives direct light from a light source or not. To satisfy the expectations for this project, when a point lies “in shadow” you do not need to consider the concept of **direct transmission** of light from a light source through a transparent medium that exists between the point and the light source.

The scene you render must contain **at least five different surfaces** (*e.g.*, plane, sphere, ellipsoid).

Besides having to hand in a program listing, please prepare a “manual sheet” explaining how to use your program.

The overall grade (on a scale from 0 to 100) will depend on i) **completeness** (40%), ii) **correctness** (40%), iii) **interface quality** (15%), and iv) the **manual sheet** (5%). No project will be accepted when it is more than seven (7) days late; for each day, one (1) point will be deduced.

HAVE FUN !!!