# Spring 2016 HW2: Ray Cast and Shading Due Date: March 14, 2016

## Description

In this assignment you will write a basic ray tracer. A substantial amount of starter code is provided in a template program called "ray" which takes care of many of the low-level details of parsing, data structures, transformations, etc., as well as containing high-level functions outlining the ray tracer. Your job is to fill in missing code in key functions to complete the implementation. Every place in main.cpp that requires additional code or some modification is marked with the comment FILL IN CODE. You may make support functions and types as necessary, but try to make all changes in ray.cpp rather than ray.cpp or ray.h. In your README any changes made elsewhere must be noted.

## Required elements for both undergraduate and graduate students

- Complete DIFFUSE section of shade_ray_diffuse()
- Complete shade_ray_local(), which adds specular.

As the last point implies, you must not only implement the ray tracer but also create a scene that shows off its functionality. Do this by creating a custom .scene file (see below for format) to arrange objects, place lights, and choose a camera angle. Submit the image that you produce as a .ppm or your favorite image format, along with your scene description file. The scene you create should demonstrate as much functionality as you implement, including diffuse shading, shadows, and reflections (to a depth of at least 2 reflections) on multiple objects and spheres. All submitted images should be rendered at a resolution of at least 500 x 500, although a much smaller image size is recommended while you are debugging.

## .scene file format

There are several kinds of commands contained in a .scene file type, each appearing on a separate line. Whole lines can be commented out by starting them with '#'. Here are the command types:

1. Camera position: **camera x y z dx dy dz upx upy upz**. Always the first command in the file; arguments work like gluLookAt().

2. Clip planes: **clip left right bottom top zNear zFar**. Always the second command; arguments work like glFrustum().

3. Image dimensions: **image width height**. Third line in file. By reducing the image size, you can debug your code without waiting too long for it to run.

4. Light(s): **light x y z amb_r amb_g amb_b diff_r diff_g diff_b spec_r spec_g spec_b**. One light per line.

5. .obj object(s): **obj path_to_file x y z sx sy sz rx ry rz amb_r amb_g amb_b diff_r diff_g diff_b spec_r spec_g spec_b shine IOR reflectivity transparency**. One object per line. (x, y, z) positions the object and (sx, sy, sz) scales it. Rotations (rx, ry, rz) are read but not currently implemented. Subsequent values parametrize material properties (overriding any that may be in the .obj file itself).

6. Sphere(s): **sphere x y z radius amb_r amb_g amb_b diff_r diff_g diff_b spec_r spec_g spec_b shine IOR reflectivity transparency**. One per line.

Here is an example .scene file (included with udray code above). It makes the following images with different shading options (these are set at end of trace_ray():