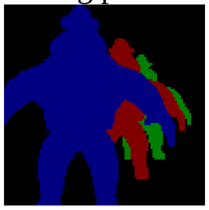
CSC 4356
Programming Assignment 4

Due 12/5/2017, 11:59pm

Objective: This programing assignment is designed to familiarize you with the ray tracing algorithm.

Requirements:
In this assignment, you are required to implement a basic ray tracer based on the skeleton code. A substantial amount of starter code is provided in a template program called "myray" 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. You'll need to fill in missing code in key functions to complete the implementation. Every place in "raytracer.cpp" (entry point to the main function) that requires additional code or some modification is marked with the comment INSERT YOUR CODE HERE. You may make support functions and types as necessary, but try to make all changes in raytracer.cpp rather than myray.cpp or myray.h. (Total 15pt + 5 extra credits)

1. Download the program package "PA4.zip" from our course website and create a project to setup the skeleton code. (2pt)
   a. add all .cpp files (raytracer.cpp, myray.cpp and glm.cpp) to the Source Files and all .h files (myray.h and glm.h) to Header Files.
   b. In Project Properties add freeglut directories to "Additional Include Directories" and "Additional Library Directories" (same as what you've done for programming assignment 1 and 2).
   c. In Project Properties – C/C++ – Proprocessor – Preprocessor Definitions add the following parameters:
      NDEBUG
      _CONSOLE
      _CRT_NONSTDC_NO_DEPRECATE
      _CRT_SECURE_NO_WARNINGS
   d. In Project Properties – Debugging – Command Arguments, put in test.scene.
   e. If you setup the project successfully, your ray tracer will render the following image: (the ray tracing process may take several minutes)



2. Complete DIFFUSE section of shade_ray_diffuse(). (2pt)
3. Complete shade_ray_local(), which adds specular and shadow effects. This function may call shade_ray_diffuse(). (2pt for specular component and 2pt extra credit for shadow effects)
4. Add sphere intersection testing in intersect_ray_sphere(). This function should parameterize the Intersection object returned (with material information, normal direction, etc.) like intersect_ray_glm_object() does. (4pt)
5. Complete reflection component of shade_ray_recursive(). (4pt)

6. Add support for refraction in shade_ray_recursive(). You may assume that there is no nesting of transparent objects (this means that spheres should be regarded as solid) and that the medium outside the objects is air. (2pt extra credits)
7. Design a complex and creative scene by modifying the .scene file (1pt + 1pt extra credits for creative scenes)
    - The scene you create should demonstrate as much functionality as you implement, such as diffuse & specular shading, shadows, reflections, and refractions on multiple objects and spheres.

What to Submit?

1. Source code: “raytracer.cpp” (or any other source files that you’ve modified)
2. Scene file: “test.scene”
3. Executable: *.exe file
4. The rendered image: “output.ppm” (output.ppm is automatically saved. The image you submitted should be rendered at a resolution of at least 500 by 500, although a much smaller image size is recommended while you are debugging. The image should match the scene file.)
5. A report explaining your implementation with key results.

.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.