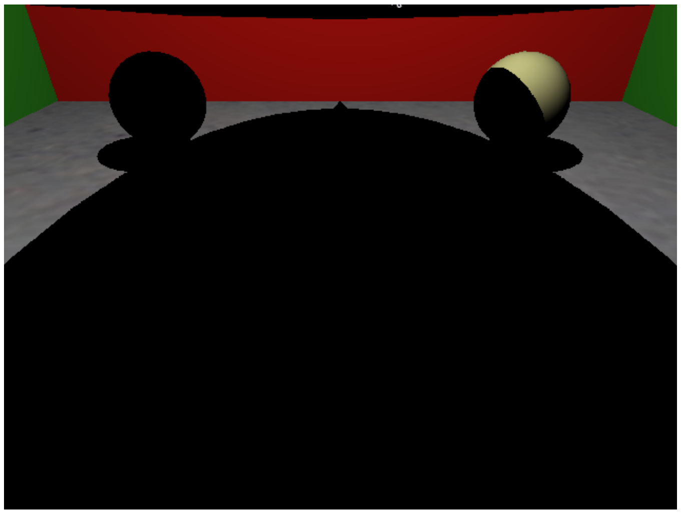
CMPT 485/829: Assignment 3 Specification
Due – Friday March 16, 2012: 11:59pm


(a) OpenGL.

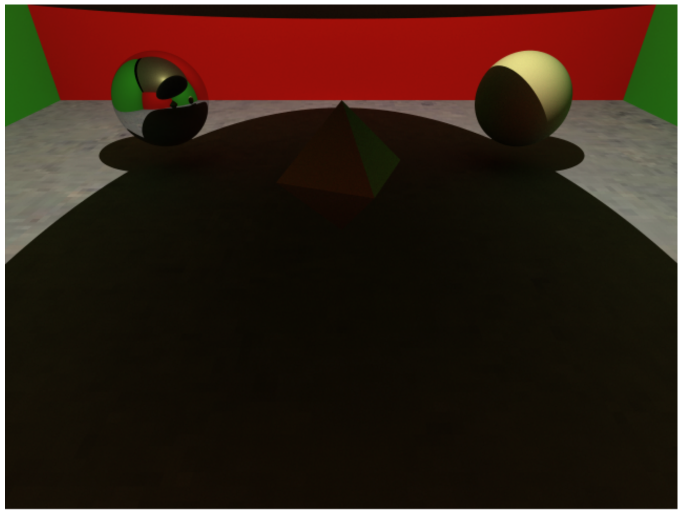

(b) Path traced to 500 rays per pixel.

Figure 1: Sample scene.

1 Introduction

This assignment is the third, and final, part of a multi-assignment mini-project that will span the term. At the very end you will have a single OpenGL program that:

1. Renders an environment in real-time;
2. Has simple camera controls to control where you are looking;
3. Renders shadows; and
4. Allows you to ray trace a view with the press of a button.

In this assignment you are going to focus on adding ray tracing (not real-time) capabilities to the program.

2 Framework

As with the first assignment, you are being provided with a C++ framework for a basic OpenGL renderer. There are a few differences between this framework and the solution provided for assignment #2:

- A new Scene object in src/scene.h has been added, and code refactored to use it.
  - This scene object contains many of the stubs that you will be required to fill in.

- Some basic functionality and stubs required for ray tracing are added. Specifically:
  - All of the shader classes now have functionality to shade a point, given material properties and such, on the CPU. This is provided through the interface class RayTracing::Shader found in src/RayTracing/rayshader.h.
  - All of the geometric objects have had functions added, for ray-object intersection, and ray-shadowing. These are provided by inheriting the interface class RayTracing::RayIntersector found in src/RayTracing/rayintersector.h.
  - Ray tracing is toggled by pressing F2; if the camera position has not moved between toggling ray tracing off and then back on, then ray tracing will resume where it left off.
  - The main loop for ray tracing is located in the idle() function of the Assignment3 class.

Note: It will become apparent in this assignment that the shadow mapping implementation provided in this framework has some bugs. The shadows that you calculate through ray tracing may not be the same as the shadows demonstrated by the shadow map.

3 Assignment 3

You have one main goal in this assignment: Implement path tracing into the framework so that the scene can be ray traced.

You may modify any code present in the framework, and add any new source files that you require. Many of the places where you will need to add code can be found by grepping the source tree for ”TODO”.

Your tasks:

1. [40 marks] Implement basic ray casting, without shading. This entails implementing:
   - Camera::setWindowToWorld()
   - Camera::genViewRay()
   - Scene::rayIntersects()
   - The todo in Assignment3::idle()
2. [20 marks] Implement the sphere-intersection functions. This includes identifying information calculated during ray-intersection that will be useful for Sphere::hitProperties()
   - See: src/Objects/Models/sphere.cpp
3. [20 marks] Implement shading without indirect illumination. This will entail implementing:
   - Sphere::hitProperties()
   - Scene::shadowRay()
   - Scene::shadeRay()
4. [30 marks] Implement indirect illumination via the path tracing algorithm. This will entail implementing:

- Ray_t::randomDirection() in src/RayTracing/ray.cpp
- Extending Scene::shadeRay() to cast an indirect lighting ray when the recursive depth remaining is non-zero, and add the radiance/color of the indirect ray to the ray shade/color.

Note: Path tracing an image to a large number of rays per pixel will take a very long time. The image in figure 1b required approximately 30 minutes to create.

3.1 CMPT 829 Students

Students enrolled in CMPT 829 have additional goals:

- Extend the material properties so that you can specify mirrors (object that reflect light in the mirror direction); and
- Extend your path tracer so that it will ray trace mirror objects.

This will entail:

- Extending the definition of Material::Material (see Shaders/material.h) to allow you to provide a mirror reflectance.
- Extend the Scene::shadeRay() function to also cast a mirror reflectance ray, and add it's color to the shade, only when the object is mirror reflecting.
- Change the material properties of one of the spheres in the sample scene so that it mirror reflects.

Note: You are not expected to implement mirror reflections in the OpenGL render-path. Simply leave the OpenGL render-path as-is so that it will ignore mirror properties (i.e. it will continue to only consider the lambertian and specular shading properties of the material).
Students enrolled in CMPT 485 may complete this part for bonus.

4 Deliverables

1. Upload your completed programming as a zip or tgz file to ”Assignment 3” on Moodle. Your source code must compile and run on the Linux machines in S360, and include a Makefile that builds the project.

5 Evaluation

The assignment will be graded subjectively by the marker with the following breakdown:

1. 25% – Code readability/clarity
2. 75% – Implemented to spec

Each component of the assignment is worth the following:

1. Basic ray casting – 40 marks
2. Ray-sphere intersection – 20 marks
3. Direct shading – 20 marks
4. Indirect illumination – 30 marks
5. Mirror reflections – 20 marks

The maximum grade for CMPT 485 students is 120 of 110 marks; the 829-only component of the assignment may be completed for bonus marks.

The maximum grade for CMPT 829 students is 130 of 130 marks.

Note: Moodle has no way to differentiate students by class in a split class. So, please look at the absolute mark value instead of the percentage grade when looking at your grade.