# CS 7490, Spring 2016

# Homework 5: Photon Mapping

**Due: 11:55pm, Wednesday, April 27, 2016**

---

## Objective

The subjet of this project is Photon Mapping. You will add two new capabilities to your ray tracer: 1) caustics, and 2) indirect illumination. You will achieve ech of these capabilities by adding stages to your renderer that emit photons from the light sources and store the photons at diffuse surfaces using a kD-tree. To simplify this assignment, we are only going to have photons emitted from point light sources. Don't bother to create photons from disk lights or spotlights.

---

## Scene Description Language Extensions

Below are the new commands that you will add to your ray tracer in order to give it new capabilities.

- **reflective $Cd_r$ $Cd_g$ $Cd_b$ $Ca_r$ $Ca_g$ $Ca_b$ k_refl**
  This command describes a new kind of surface material. Just as with the **diffuse** command, this command defines the diffuse and ambient color components of a surface. In addition, however, it also defines a reflectance coefficient **k_refl**. This value should be in the range of zero to one. When it is non-zero, **k_refl** indicates how much of the light that strikes the surface is to be reflected. Eye rays that hit such a reflective surface should spawn a new reflective ray, and this reflective ray should contribute to the surface's color. Moreover, if a caustic photon hits such a reflective surface, this should cause the caustic photon to "bounce", and to travel in a new direction. Caustic photons only stop bouncing when they hit a diffuse surface.

- **hollow_cylinder radius x z ymin ymax**
  Create a hollow cylinder that has its axis parallel to the y-axis. The hollow cylinder should not have end caps.

- **caustic_photons num_cast num_near max_near_dist**
  This command indicates that each light source will cause **num_cast** number of caustic photons to be shot into the scene. Each of these photons will carry a fraction of the power of each light source into the scene. Caustic photons will bounce off reflective surfaces, and will hit and be stored at diffuse surfaces. You will use the provided code for a kD-tree to store these photons. When calculating the shading for diffuse surfaces, you will query the kD-tree of caustic photons to find the nearest **num_near** photons. The power of these nearby photons will add to the amount of light falling on this surface. The kD-tree routine for finding nearby photons needs to know a maximum radius in which to search. The parameter **max_near_dist** is this parameter.

- **diffuse_photons num_cast num_near max_near_dist**

This command indicates that each light source will cause **num_cast** number of global illumination photons to be shot into the scene. Each of these photons will carry a fraction of the power of each light source into the scene. You will use provided code for a kD-tree to store such photons. Like caustic photons, these photons will be stored at diffuse surfaces. Unlike caustic photons, these photons will often bounce off diffuse surfaces, with a probability that is proportional to their diffuse color coefficients (Russian roulette). Note that you do not store diffuse photons at the first diffuse surface that they hit, only at the second and later surfaces. When calculating the shading for diffuse surfaces, you will use these photons to estimate the indirect illumination diffuse surfaces. Do this indirect estimate in the same manner as the caustic photons. A fully implemented photon map system would use a "final gather" to estimate incoming irradiance, but we are skipping this step in order to keep the difficulty of this assignment reasonable. This means our results will be more noisy than a renderer that uses final gather.

- **final_gather num_rays**
  This command is no longer part of this project. We are not going to implement the "final gather" stage of photon mapping.

---

## Provided Code

We provide source code (photon.pde) that contains kD-tree routines. You should use this code to store your photons at diffuse surfaces. Note that the photon class that is provided is minimal, and in particular does not hold the power of a given photon. You should add this information to the photon class.

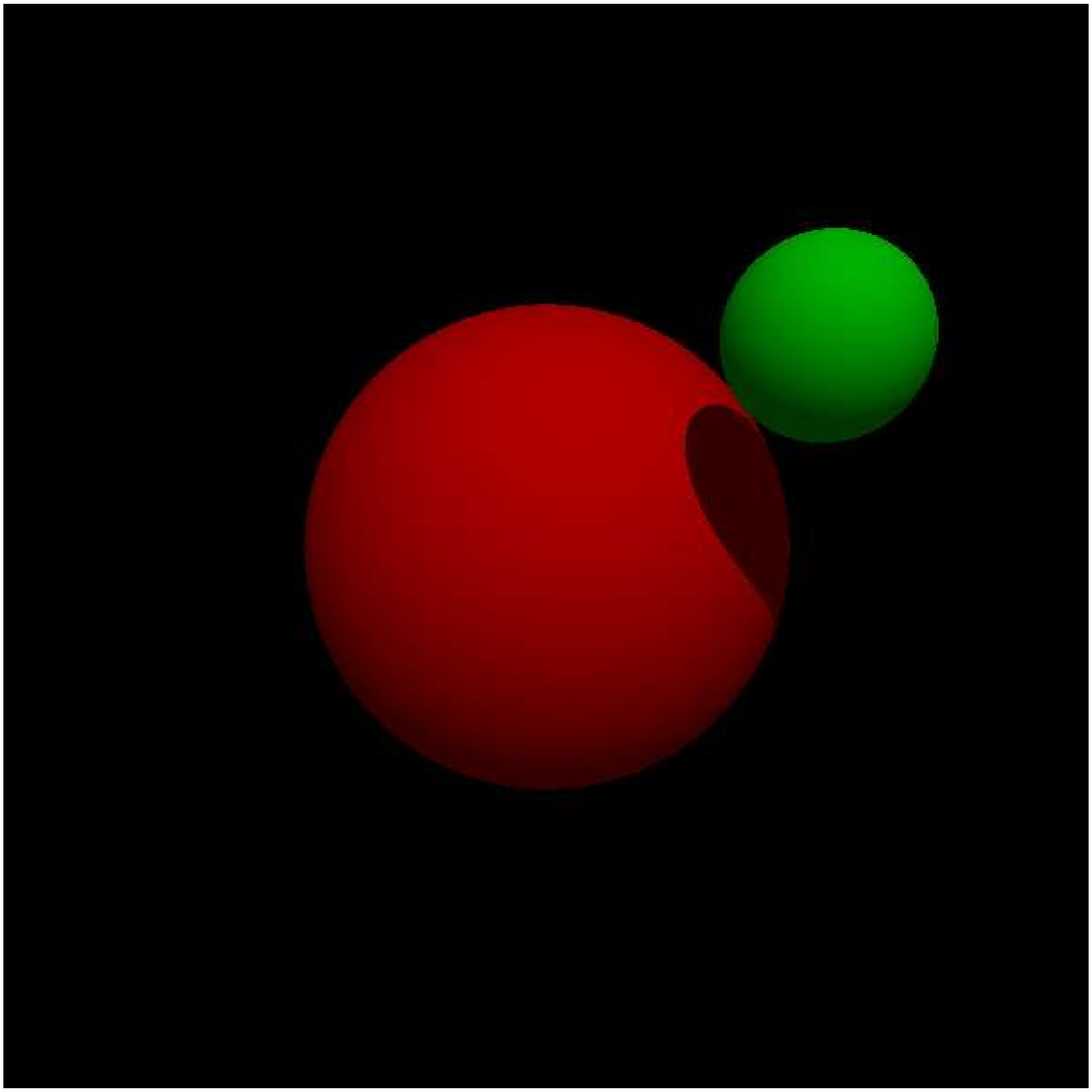
This provided code comes as part of a simple visual demo of the kD-tree class. The demo draws many black points on the screen, and stores these points in a kd-tree. Using the mouse position as a query location, this demo searches for the k-nearest points, and draws these nearby points in red.

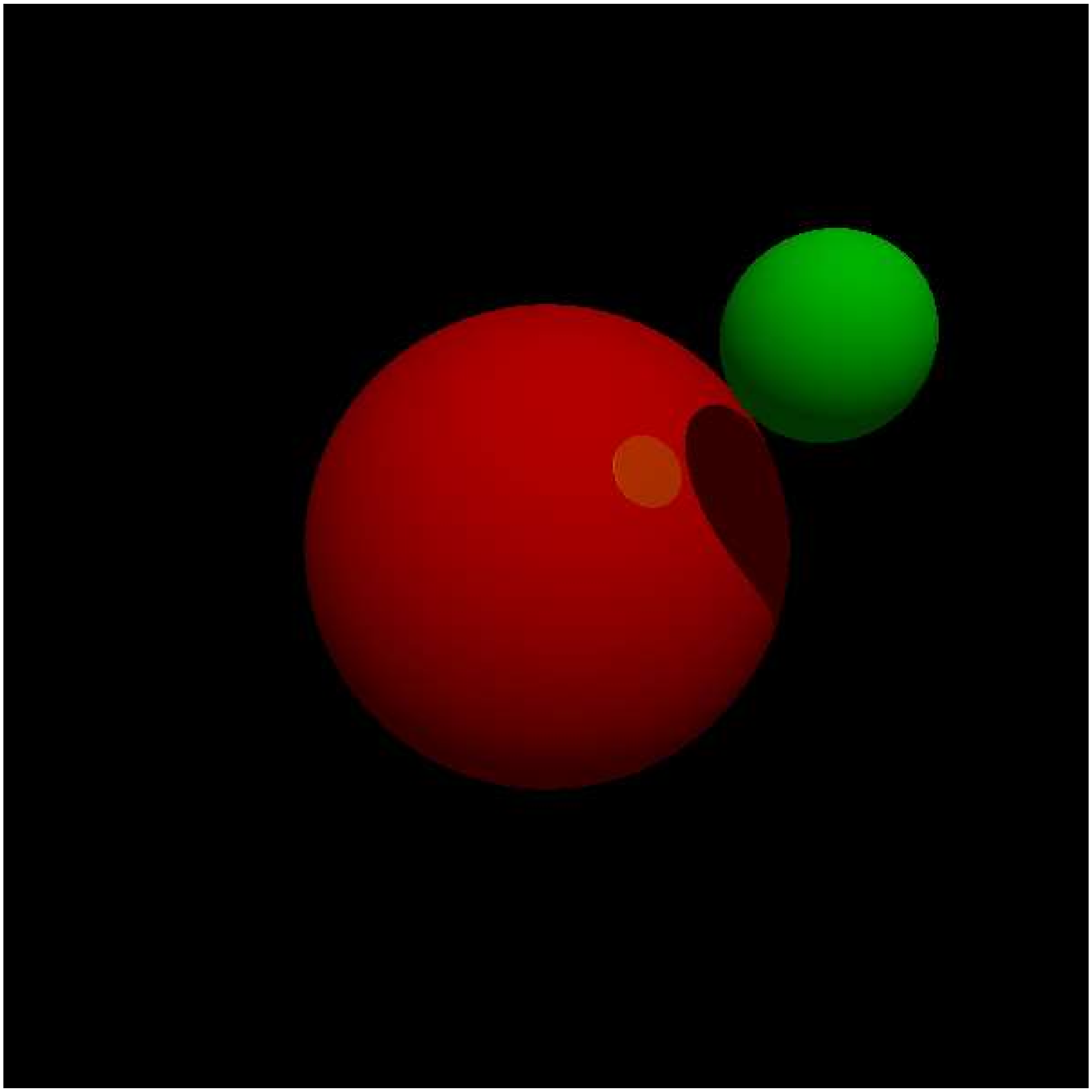
---

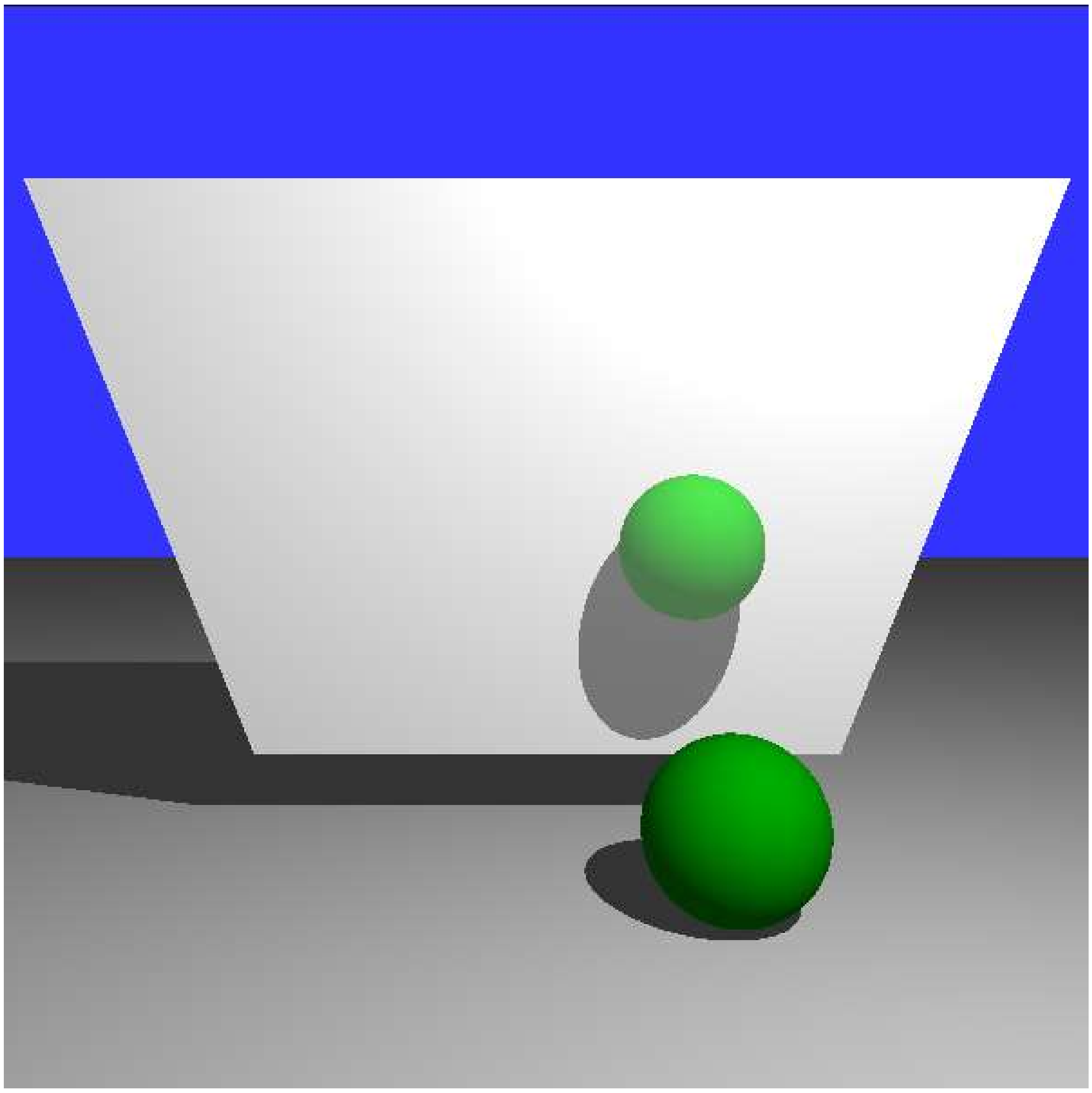
## Scene Files

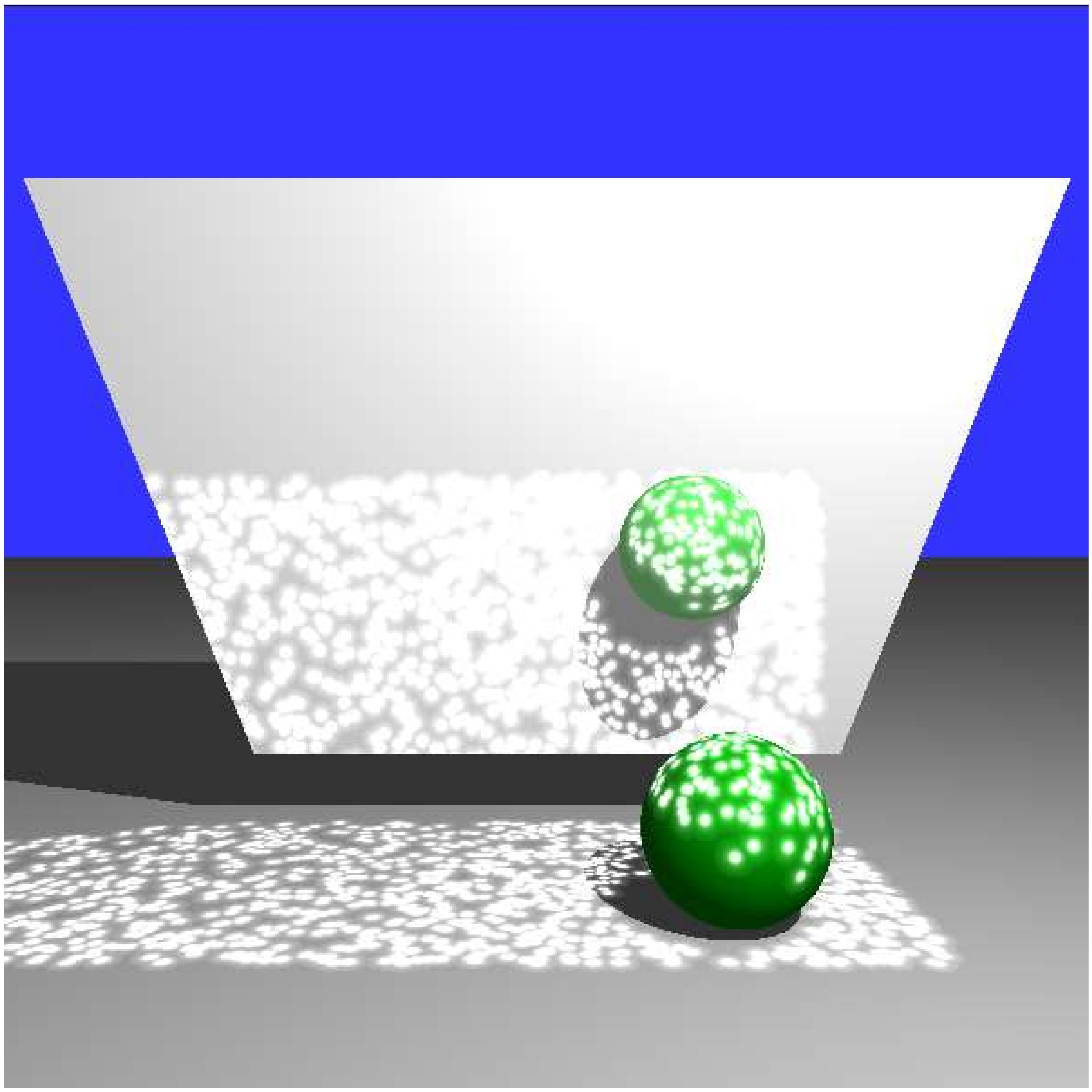
In the directory "**data**" are several test scenes that are described by .cli files. Also in that directory are the images that should be created by these scene files. Please note that I am not yet done implementing the diffuse photon mapping, so the examples for this are not yet ready. Check back in a few days for more .cli files for global illumination. The provided source code, CLI and image files are in this zip file.

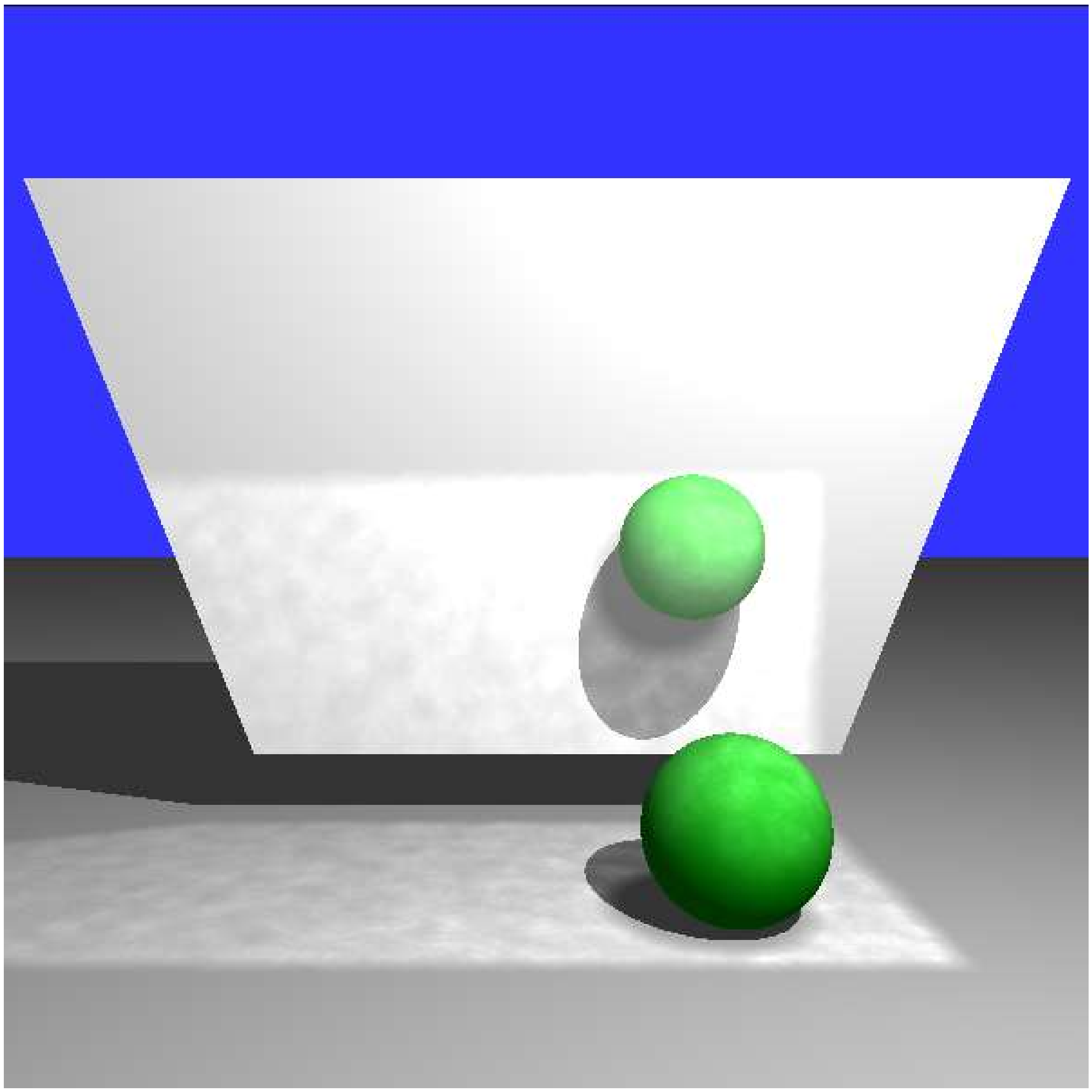
**Sample Results:**

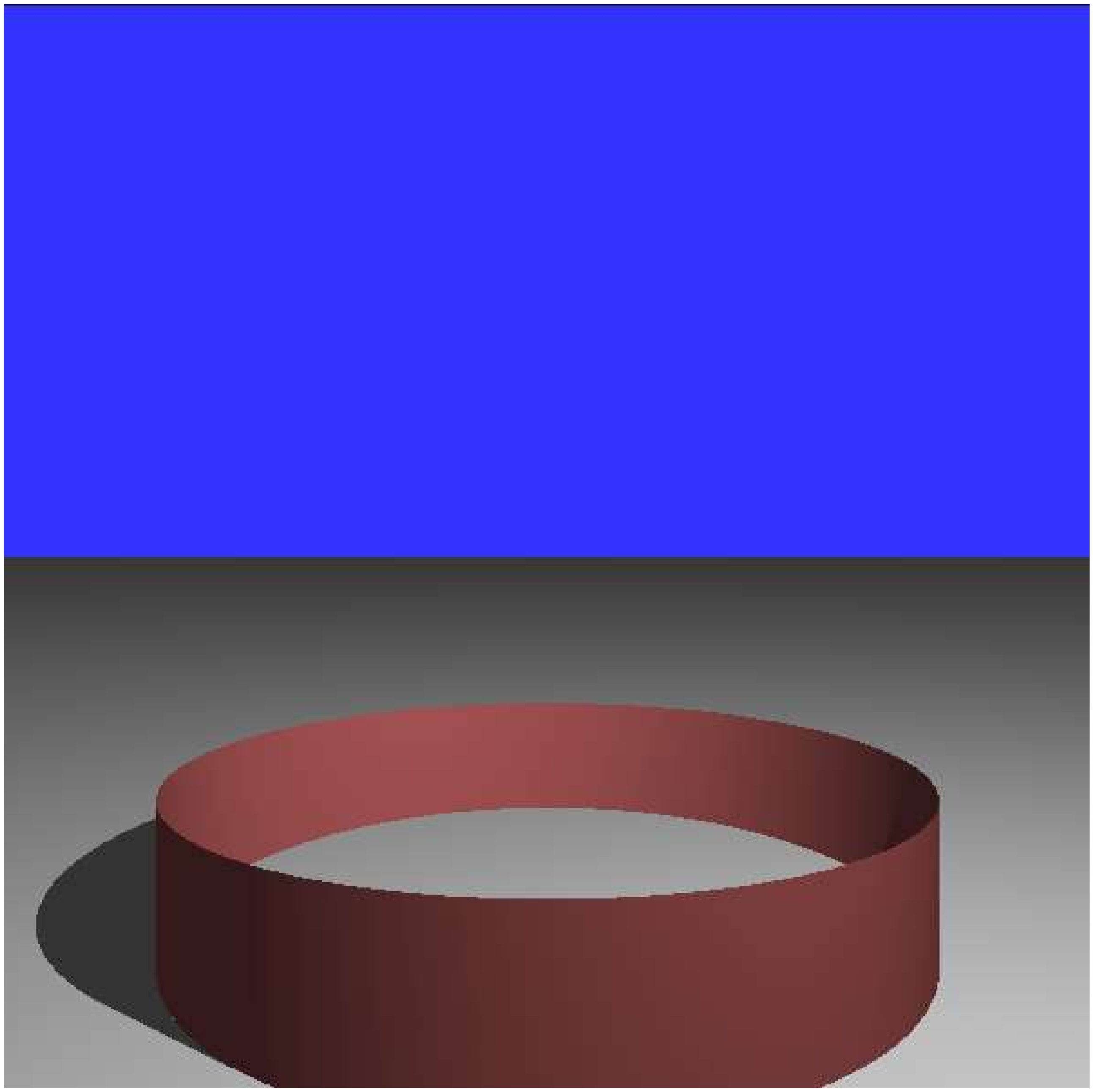


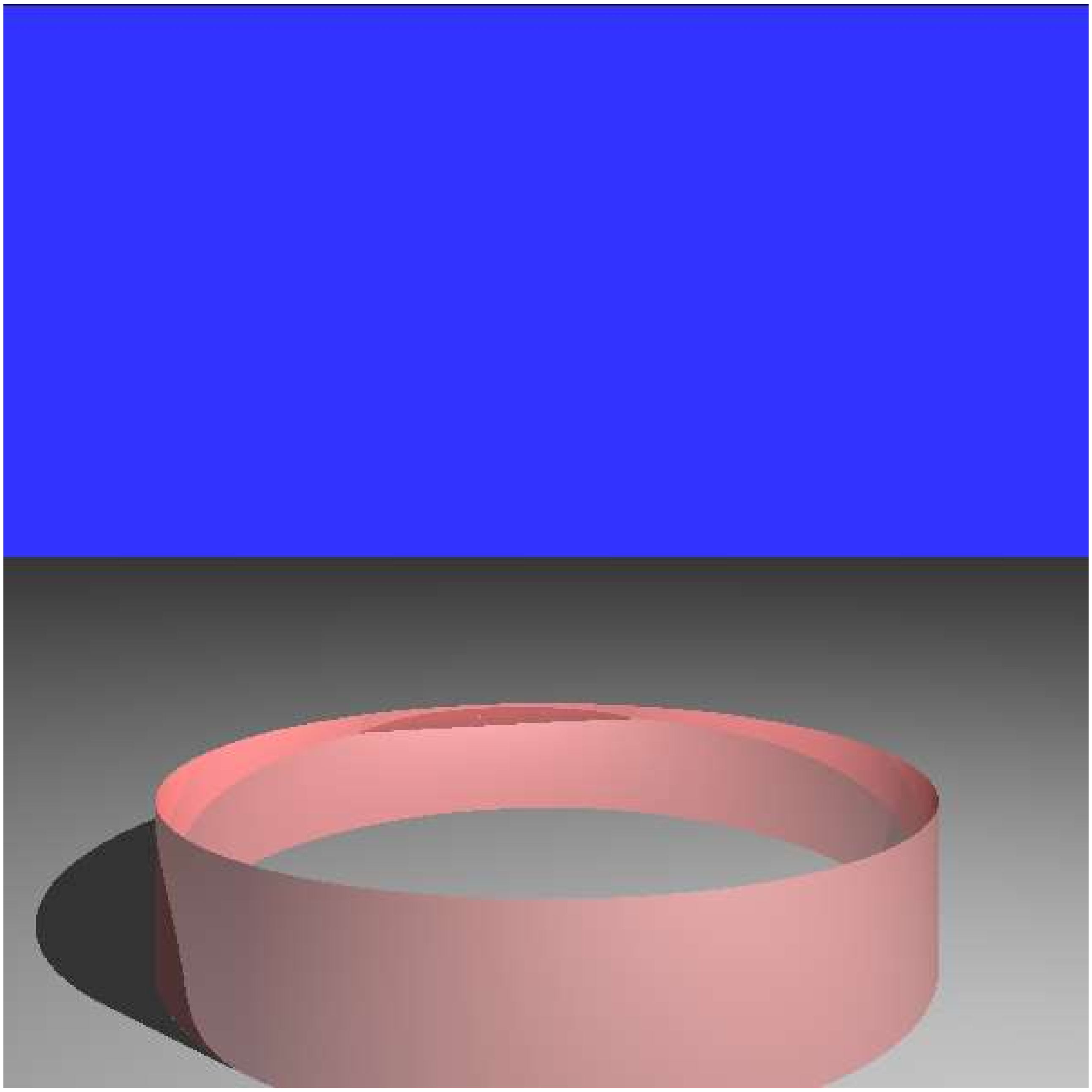


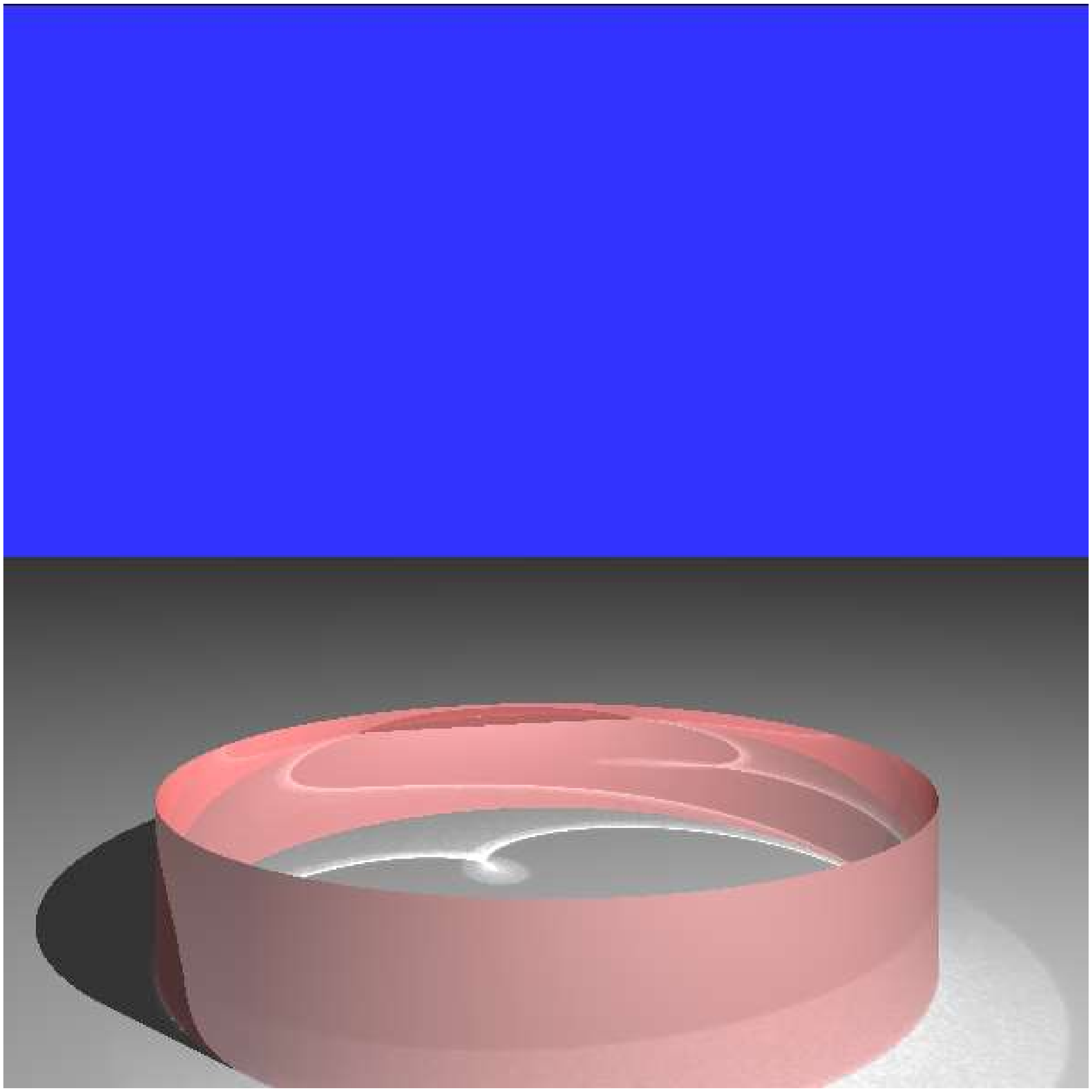


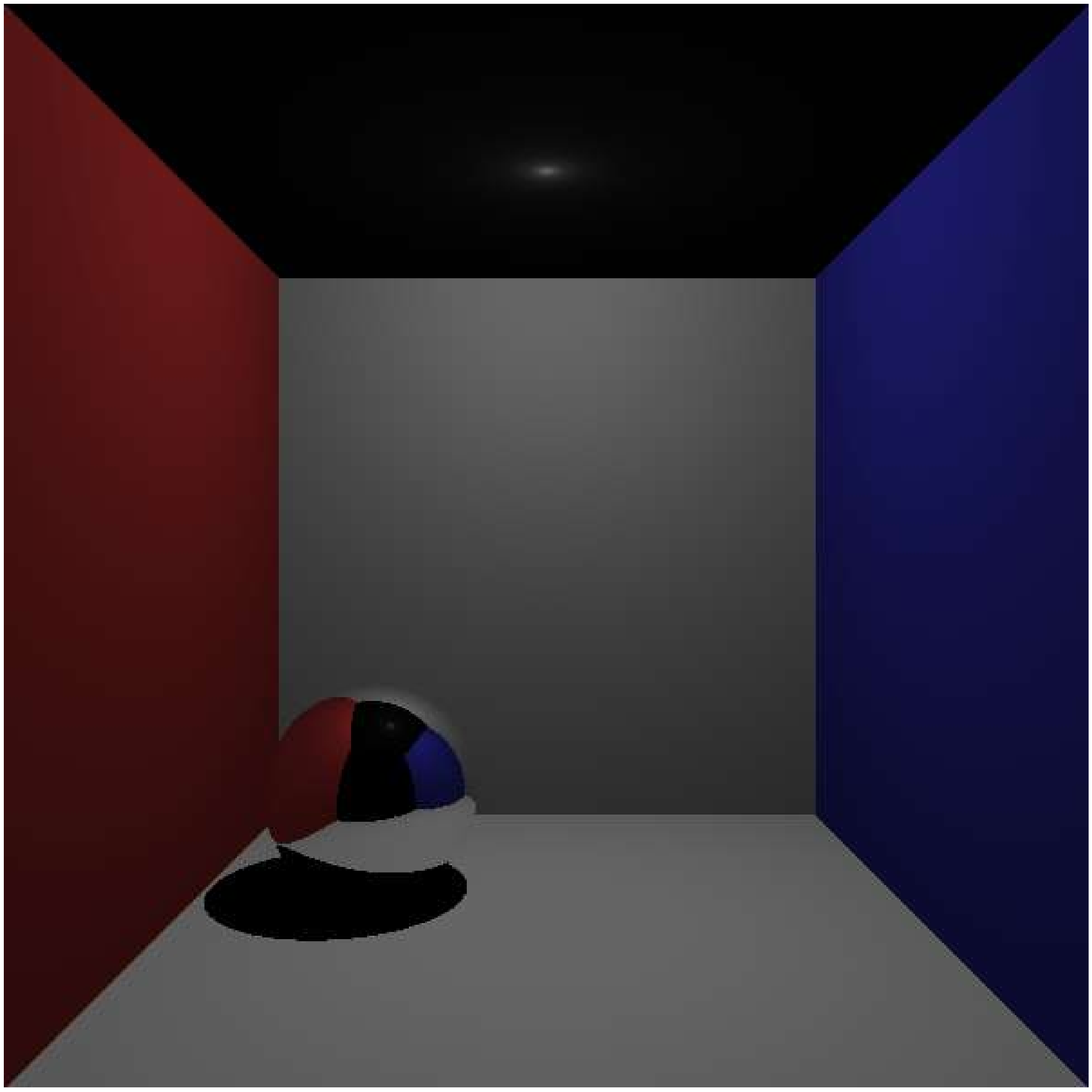












## Suggested Approach

First, you should add the **reflective** command to your ray tracer, and make sure that you can create surfaces that show reflections in them.

Next, begin on caustic photons, testing it on the scene that contains a single square reflective surface. You will need to add a caustic photon tracing routine that is called before you render anything else in the scene. This new routine will create many photons for each light source and trace them throught the scene. (Note that we will only create photons from point light sources, and that you don't need to create them for disk lights or spotlights.) The caustic photons will bounce off reflective surfaces, and be stored at the first diffuse surface that they hit. When calculating the final shading for a diffuse surface, these caustic photons will contribute to the final intensity of the surface. To debug this, I recommend that you first just color a point on a diffuse surface bright white if it is close to a stored caustic photon. After you are sure that the photons are being stored and found at the correct location, then you should modify this to calculate the appropriate contribution of the caustic photons, based on the power that they carry.

Once you have finished caustic photons, you might want to implement the hollow cylinder. This will then allow you to properly render the scene that creates a so-called cardioid caustic that comes from reflections off a shiny ring.

Finally, begin work on diffuse photon tracing. The routine to create these will be quite similar to caustic photons, but they will bounce off diffuse surfaces. Only store diffuse photons at the second and later surfaces that they encounter, not at the first surface. Once these are created, you should then implement estimation of the local photon density from these diffuse photons. This is the final part of this project. Because of the difficulty of this assignment, we are not going to perform the "final gather" step of photon mapping. The files t09.cli and t10.cli are the examples of without and with diffuse photons.

---

## Authorship Rules

The code that you turn in must be entirely your own. You are allowed to talk to other members of the class and to the instructor about high-level questions about the assignment. You may not, however, use code that anyone other than yourself has written. Code that is explicitly **not** allowed includes code taken from the Web, from books, or from any source other than yourself. The only exception to this rule is that you should use the source code that we provide (such as the kd-tree routines). You should not show your code to other students. If you need help with the assignment, seek the help of the instructor.

---

## Development Environment

You must use the Processing language which is built on Java. Be sure that you are using Processing version 2.0 or higher. The best resource for Processing language questions is the online or offline Processing language API (found in the "reference" subdirectory of the Processing release).

---

## What To Turn In

Compress the whole folder for this project (not merely the files within the folder) into a zip archive submit them to T-square. The zip archive should be included as an attachment. When unzipped, this should produce a folder containing all of your .pde files and a directory "data".

---