CS 7490, Spring 2016

Homework 2: Distribution Ray Tracing

Due: 11:55pm, Monday, February 22, 2016

Objective

This project's emphasis is on producing various soft effects in rendering through distribution ray tracing. The effects that you will produce include: soft shadows, motion blur, and depth of field. You will use your Project 1 ray tracing code as a starting point for this assignment.

Scene Description Language Extensions

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

- **rays_per_pixel num**
  Specify an integer that tells the number of rays that should be shot per-pixel to make an image. When just a single ray is shot, the ray should pass through the center of the pixel. If multiple rays are to be shot, however, each should pass through a random position within each pixel. These multiple rays for a given pixel should be averaged together in order to give the final color for the given pixel. When shooting more than one ray per pixel, this should result in anti-aliasing at the pixel level, thus turning stair-step aliasing into more smooth edges. Note that the time it takes to render your images will be proportional to this ray replication factor. A value of rays_per_pixel of 5 will take five times as long to render as a value of 1.

- **disk_light x y z radius dx dy dz r g b**
  Create a disk-shaped light source. The center of the disk is (x, y, z), the radius is rad, and a normal vector to the disk is (dx, dy, dz). As with point lights, there is also a color associated with the light (r, g, b). Since this light source has a non-zero area, it sould cast a shadow that is soft on the edges. For each shadow ray that is cast at this light source, you should select a random position on this disk.

- **moving_sphere radius x1 y1 z1 x2 y2 z2**
  Create a sphere that is travelling linearly between the position (x1, y1, z1) and (x2, y2, z2) while the scene is being rendered. Because of this motion, the sphere should be motion blurred. Each ray that is tested against this sphere should be assigned a random time in the interval [0,1], and the sphere's position should be determined by this random value.

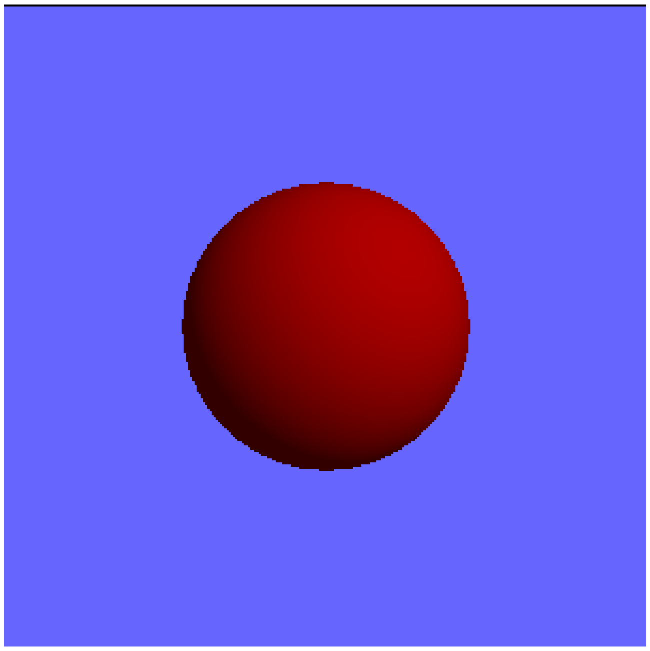
- **lens radius focal_distance**
  This command specifies the lens size (radius) and what distance in front of the eye is in focus. You should use this information to create eye rays that go through random positions on the lens and that are aimed at particular positions on the focal plane. Objects on or near the focal plane should appear in focus, and objects that are away from the focal plane should

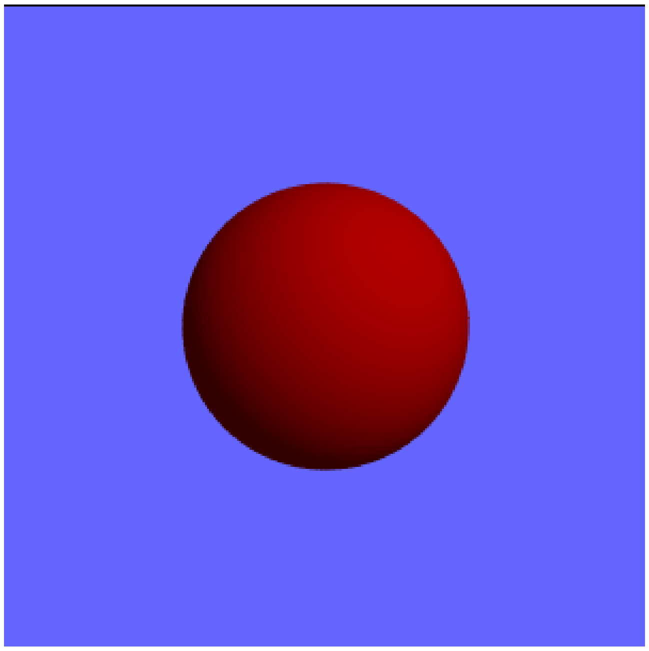
be out of focus. A lens with a larger radius will create more blur.

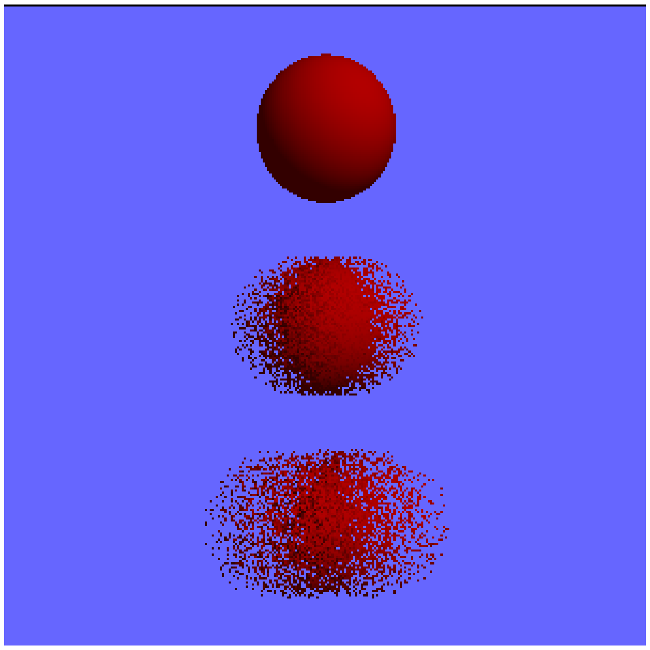
Scene Files

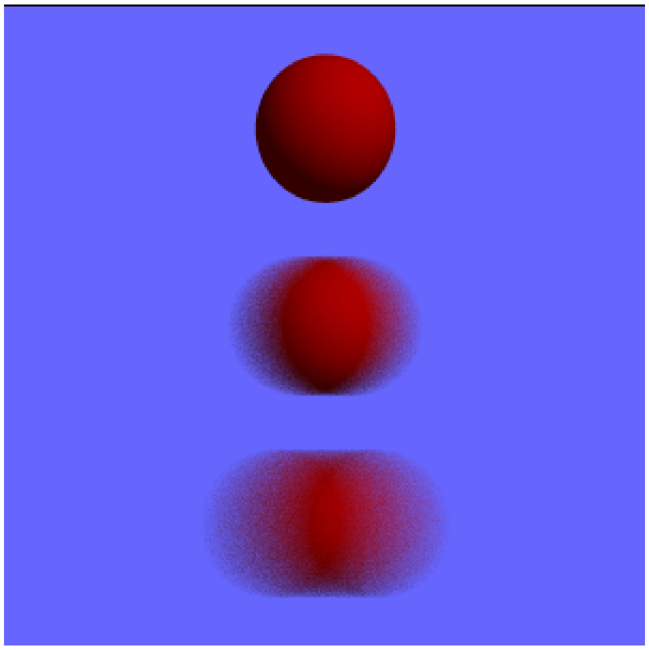
In the directory "**data**" are several test scenes that are described by .cli files. Also in that directory are the images that should be created by these scene files. The provided CLI and image files are in a zip file. Pressing keyboard keys 1-9 and 0 calls these test CLI files.

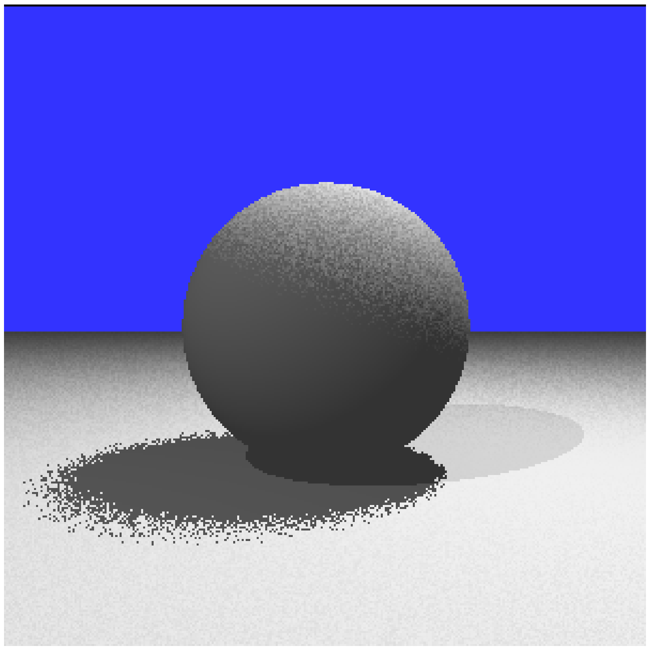
Sample Results:

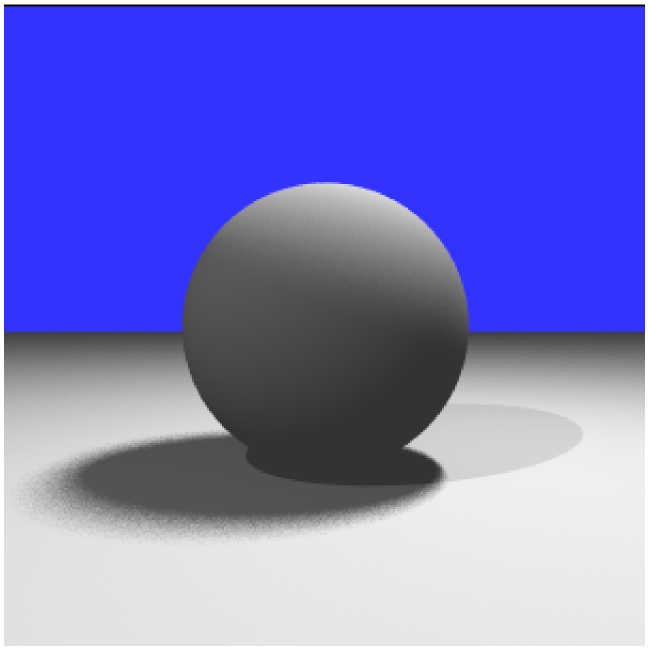


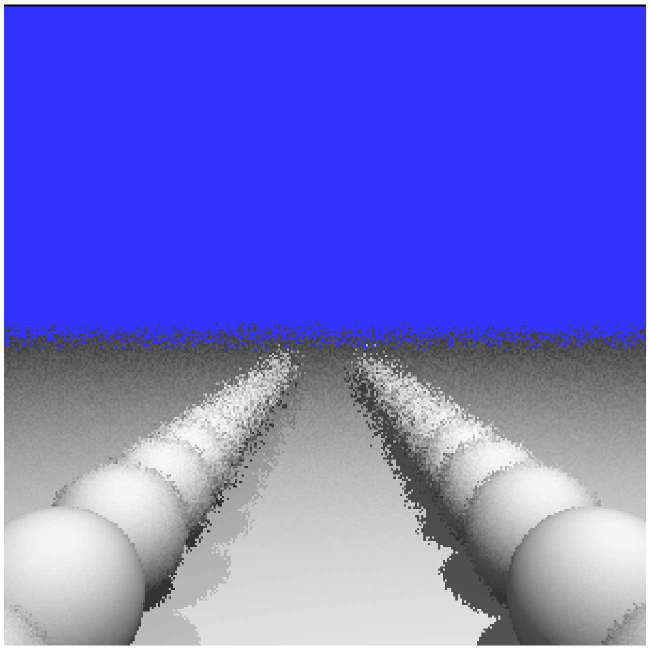


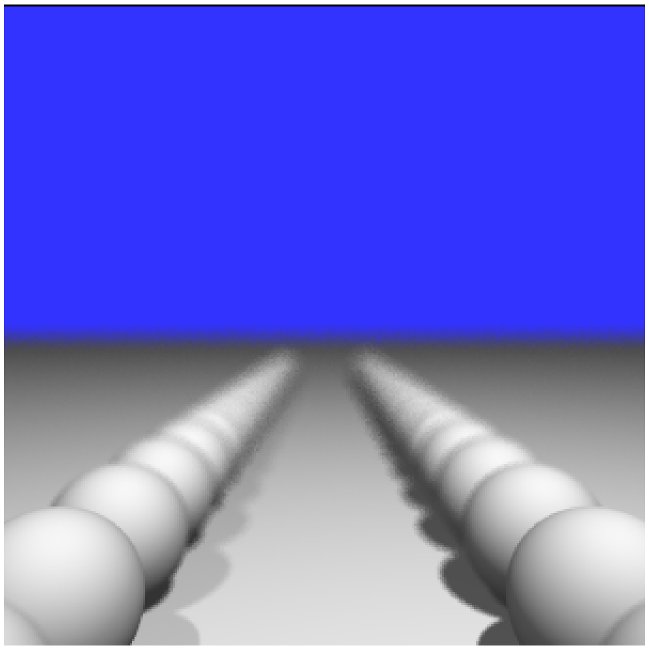


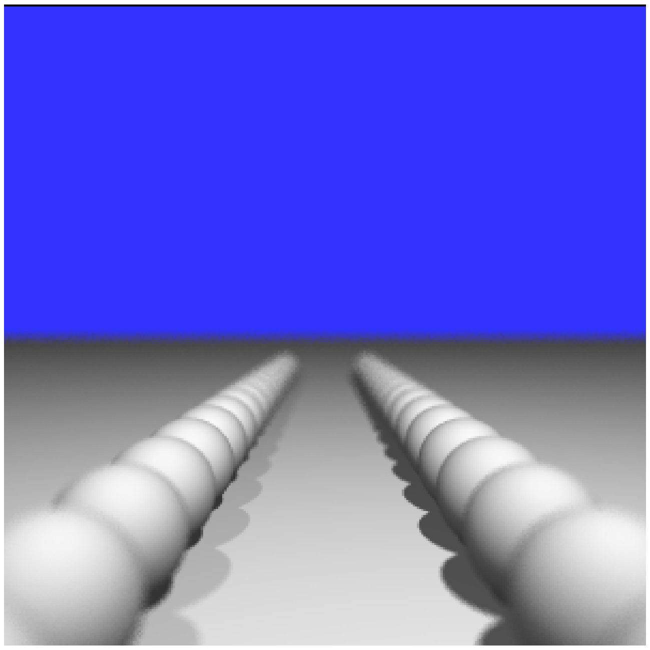












Note that the results for t01 and t02 are different, but the differences are very subtle. The border of the sphere in t02 are anti-aliased due to multiple samples per pixel, while the border of the sphere in t01 are stair-stepped due to aliasing. Performing image zoom on these results will make the differences more obvious.

Suggested Approach

It is important that you implement rays_per_pixel first, since all of the other effects depend on shooting and averaging together multiple rays. Once you have multiple rays per pixel implemented, then the other effects (motion blur, depth-of-field, soft shadows) can be done in any order.

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 code from project 1. 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".
