Project 3A: Ray Tracing Spheres

Due Oct 24 by 11:59pm **Points** 100 **Submitting** a file upload
Available Oct 5 at 11:40am - Oct 27 at 11:59pm

This assignment was locked Oct 27 at 11:59pm.

Objective

This is a very difficult assignment! You should start on this project way before the due date, to give yourself time to write and debug your code.

The goal of this project is to write a ray tracing renderer. You will write a collection of python functions that, when called, will create a 3D scene and produce 2D images of the scene. One of the functions will initialize the scene, others will create objects, lights and a virtual camera, and one additional function will cause the scene to be rendered into a 2D image. You will be provided with various scenes that will call your functions to test your ray tracing code.

This is the first half of a two-part project. For this first part you will cast eye rays into the scene for each pixel, test these rays for intersection with spheres, and then use the diffuse shading equation to find the color for each pixel. In the second half of the project you will expand your Ray Tracer to include new functionality.

Project Description

You have four primary goals for the first part of this project:

1. Initialize the scene
2. Cast eye rays for each pixel
3. Implement detection of ray intersection with spheres
4. Implement the diffuse shading equation

You can accomplish these goals however you see fit. A good approach would be to use object oriented programming practices and create objects for each of the major scene components (light, ray, surface material, sphere). Global lists of scene objects can be stored to allow for easy access. You should then create a ray object for each pixel. To debug your eye ray creation routine, calculate by hand what the ray should be for a pixel in the middle of the screen, and compare this to what your code creates.

The next step should be to write a method which will take a ray as input and test it against a sphere for intersection, and return the closest hit point. This Hit object, containing all necessary information, can then be passed to a shading function which would implement the shading equation (for multiple light sources) and return the pixel color. Such an approach would be easy to test and to extend in the next stage of the project.

However you implement the ray tracer, your results should appear almost exactly like the examples shown below in Results.

Provided Code

The **PROVIDED SOURCE CODE (https://gatech.instructure.com/courses/264908/files/35742565/download)** **(https://gatech.instructure.com/courses/264908/files/35742565/download?download_frd=1)** contains functions that describe various scenes with spheres. Each number key 1-4 is assigned to a single scene function and pressing that key should reset the scene and create an image of the new one.

You should modify the source code in any way you see fit, and comment your code (include placing your name in the header). The source code is written in the Python version of Processing. Visit “py.processsing.org/reference/” for more information on built in functions. You are **NOT** allowed to use most of the built in processing/openGL functions in this project such as for drawing spheres. We are not using rasterization for this project! You are, however, encouraged to use the PVector data type and the standard math functions. When in doubt about what library functions are okay to use, please ask the instructor or a TA.

You must set the color of each pixel manually. The easiest way to set a pixel’s color is to use the set() function, as is shown in the provided code.

Scene Description Language

Each scene is described by a text file with suffix ".cli", which stands for Command Language Interpreter. Each of these .cli files are just plain text files, and you are encouraged to look at them! We have provided code that parses these simple scene description files. You must modify this parser so that it calls your own scene setup and rendering routines.

Below are the commands that the parser and your code will need to handle for this assignment:

background r g b

Sets the background color. If a ray misses all the objects in the scene, the pixel should be given this color.

fov angle

Specifies the field of view (in degrees) for perspective projection.

eye ex ey ez

Set the position of the eye to (ex, ey, ez).

uvw ux uy uz vx vy vz wx wy wz

Specifies the **(u, v, w)** coordinate frame that helps to define the eye rays.

light x y z r g b

Create a point light source at position (x,y,z) and its color (r, g, b). Your code should allow any number of light sources. For the second part of this assignment, you will cause these lights to cast shadows.

surface dr dg db ar ag ab sr sg sb spec_power k_refl

Specifies a new surface material with diffuse color **(dr, dg, db)**. Ignore the other terms for now, since we will not use them until the second part of this assignment.

sphere radius x y z

Specifies the creation of a sphere with its center at (x, y, z) and with a given radius. The material properties of the surface are given by the last **surface** command that was executed prior to this command.

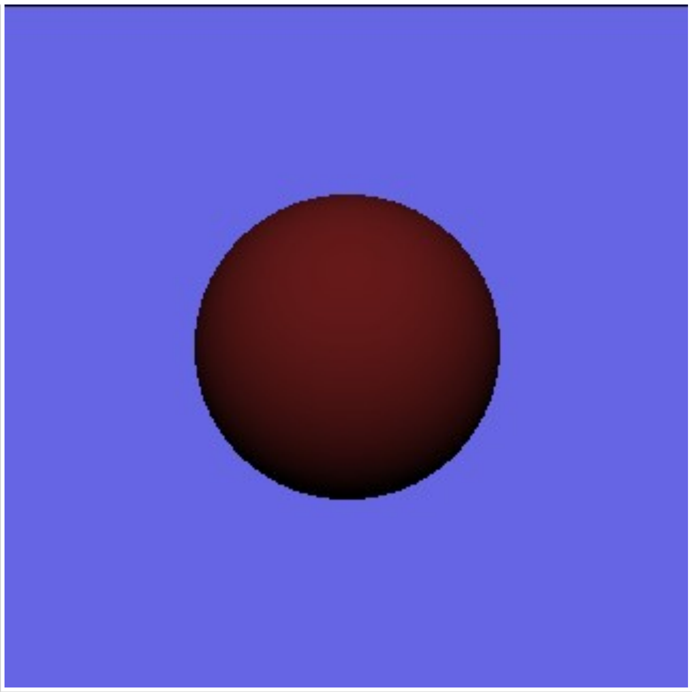
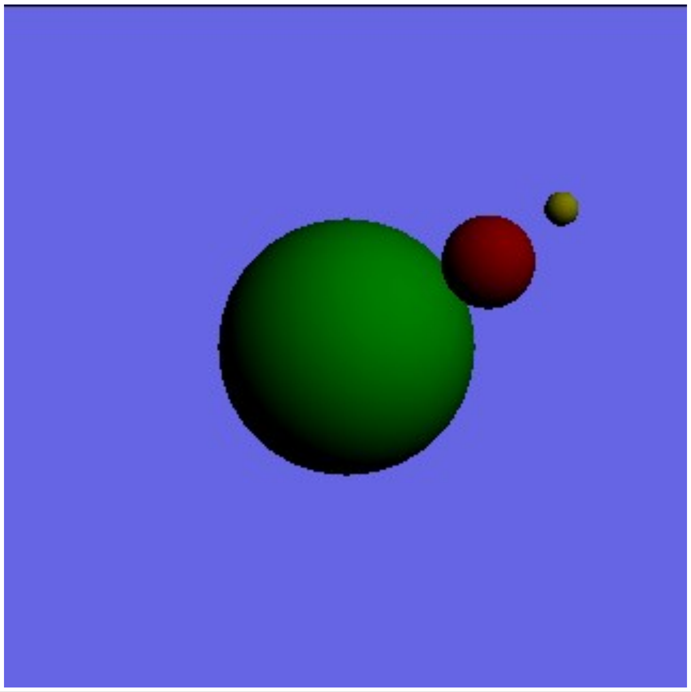
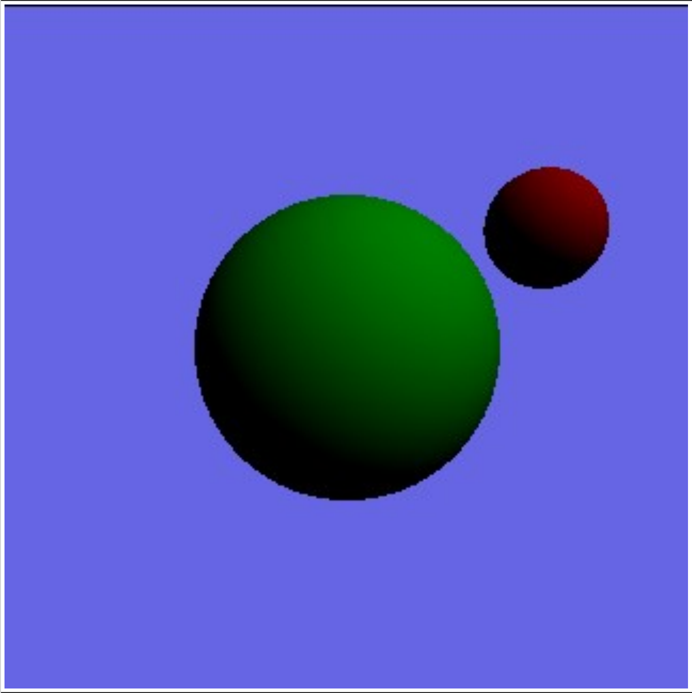
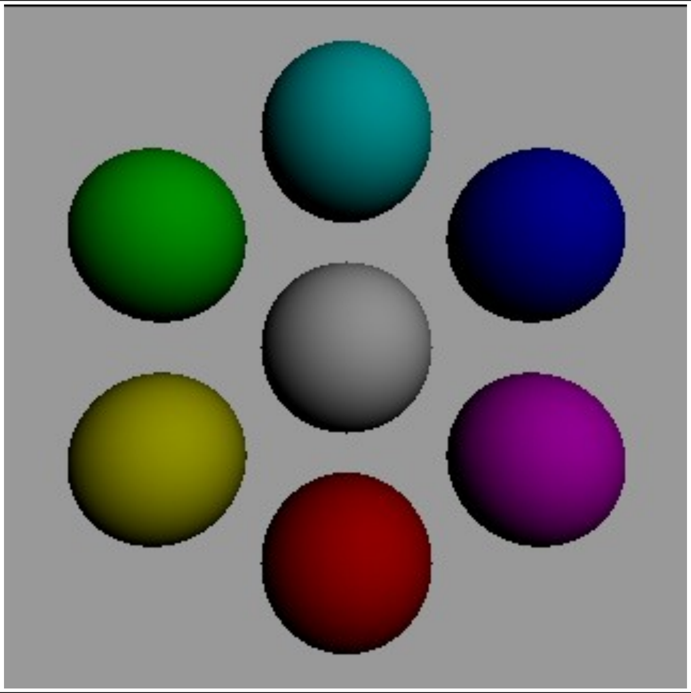
render

Ray-traces the scene and displays the image in the drawing window.

Note on color specification: Each of the red, green, and blue components for the above commands are floating point values in the range of 0.0 to 1.0.

Results

Below are the images that your program should generate for a given scene, when you press the keys 1-4. There will be additional scene descriptions provided for Part B of this project. (You can find clues about Part B by looking at the text of these other scene descriptions.)

Authorship Rules

The code that you turn in entirely your own. You are allowed to talk to other members of the class and to the instructor and the TA's about general implementation of the assignment. It is also fine to seek the help of others for general Processing/Javascript programming questions. You may not, however, use code that anyone other than yourself has written. The only exception is that you should use the example source code that we provide for this project. Code that is explicitly **not** allowed includes code taken from the Web, github, from books, from the py.processing.org web site, from other assignments, from other students or from any source other than yourself. You should not show your code to other students. Feel free to seek the help of the instructor and the TA's for suggestions about debugging your code.

Submission

In order to run the source code, it must be in a folder named after the main pyde file. When submitting this assignment, keep your code in this folder (including the "data" sub-folder), zip up this folder, and submit this one zip file via Canvas. Please do **not** use tar or rar to turn in your files.