Exercise #5 in Computer Graphics
Even Better Boxes world

Due: Last day of class

1 Introduction

This exercise adds a few features to the boxes world that you implemented in Exercise #3. Your main purpose is to implement and use a ray-tracer. In addition to the description of the boxes and cameras, your program should also handle the existence of mirrors in the world. If you are either a graduate student, or an honor student, your program should also handle lamps. Their descriptions appears below. A nice interface to control the location of the mirrors (and the lamps, if exists) is required.

It is not unlikely that you can find on the WWW pieces of code that might help you reach the goal. Yet you are expected to implement all of it by yourself, (together of course with your partner) Hence a few students might be asked to meet with the instructor or the TA, and explain their implementations.

2 Mirrors

Each mirror has a color, and is a triangle-shape. Your program should look for a file called `mirrors.dat`, which contains a few lines of the form

$$x_1, y_1, z_1,\ x_2, y_2, z_2,\ x_3, y_3, z_3,\ k_r, k_g, k_b$$

The vertices of the mirror lie in the points (x_1, y_1, z_1), (x_2, y_2, z_2) and in (x_3, y_3, z_3). Both sides of the triangles defined by these vertices are mirrors with the same properties. The factor k_r is a number (float) between 0 and 1. It specifies how much of the red light that reaches the mirror, would be reflected. The parameters k_b and k_g are defined analogously for green and blue. So for example if $k_r = 1, k_b = 0$ and $k_g = 0$, the mirror reflects only red light. If you prefer to take the "easy" (relatively speaking) of the exercise, you might assure ambient light model.

Your program would be evaluated based on its speed, so an efficient ray tracer is in place. A ray of light will not be reflected in more than 3 mirrors. You should not care about the effect of illumination on boxes caused from the mirror. However, you should show the pictures with the images of the boxes that they reflect.

3 Lamps — for graduate students and honors

In this case your program should also look for a file `lamps.dat`, each with the form

$$x, y, z,\ I_r, I_g, I_b$$

where x, y, z is the location of the lamp, and I_r, I_g, I_b is the intensity in the red, green and blue. Assume only diffuse light, with the existence of some weak while ambient light.

4 World description - as appeared in ex #3

The world box consists of mirrors, as described above, cameras, and (surprise) boxes. You'd be asked to describe the view each one of these cameras sees.

There are several coordinate systems. The universe coordinate system and the ones as defined by each of the cameras.

The program is interactive. It waits for an input from the user, given as a command from the prompt, and it reacts to these commands.

The file "boxes.dat" consists of several lines. Each line describes a box. It consists of 9 numbers:

$$x_1, y_1, z_1,\ x_2, y_2, z_2,\ R, G, B$$

where x_1, y_1, z_1 and x_2, y_2, z_2 are the coordinates of the lower left closest corner of a box, and x_2, y_2, z_2. I.e. $x_1 < x_2,\ y_1 < y_2, z_1 < z_2$. R, G, B are numbers between 0 and 1, and describes the box's color. Later on, we refer to these boxes, as box i, which is the box described in line i of the file.

We also represent 1-4 cameras. Their location is described in the file "cameras.dat". Each line is of the format $(x_1, y_1, z_1, \{p|q\})$, describing the location of the camera, and the character 'p' or 'o' describes whether this camera uses orthogonal or prospective projection. The cameras always look at the origin $\mathbf{o}_{un}$ of the universe coordinate system. Their view should include at least a piece of each box. Each camera i defines its own coordinate system as follows: The origin o_i of the i-th coordinate system is in the location of the camera. The z-direction is the direction towards the center of the universe coordinate system. Define h_i as a plane passing thorough $\mathbf{o}_i$, and orthogonal to the vector $\mathbf{o}_i - \mathbf{o}_{un}$. The y-direction is the projection on h_i of the y-direction of the universe.

Commands:

view i Show the view of the boxes from the i'th camera

pick i In the following commands, use the coordinate system of the ith camera.

move j **z d** Move the jth box in the z direction by d units. A nice animation would be appreciated. Note that the z-direction is with respect to the camera which was picked. Note that boxes might overlap. This is a common phenomenon among boxes, and should not bother use. boxes are opaque.

move j **x d** -analogous to move in the x direction.

move j **y d** -analogous to move in the y direction.

reset — move all boxes to their original location, as specified.

Comments

- Note the different between "view" command and "pick" command — they might refer to different cameras.
- Make sure to print appropriate error messages if a command is invalid.

- The use of illumination and any other tricks for enhancing the three dimensional feeling is encouraged.
- Add a document describing the good things that your program are doing.
- Submitting in groups of two (at most).
- We will spent some time checking copying between projects.
- Example files would be available on the web. We would also published extra information, corrections, and FAQ if needed.