CSC 251: Graphics - Spring 2013:

Assignment 1: The Mirror World

Drawing in 2D with OpenGL

Due: Wednesday, January 16, 7pm.

1 Goal

The goal of this assignment is to get yourselves familiarized with OpenGL as well as to brush up your linear algebra basics.

2 The World

The world is 2D containing a set of line segments that are either blocks (opaque) or mirrors (one shiny side) for light rays in the world. There is a projector in the world that throws a fan of light rays into the world (see Figure 1). The projector is modeled as a point light with a line segment containing a set of n equidistant holes (pixels) that determines the angle of each light ray.

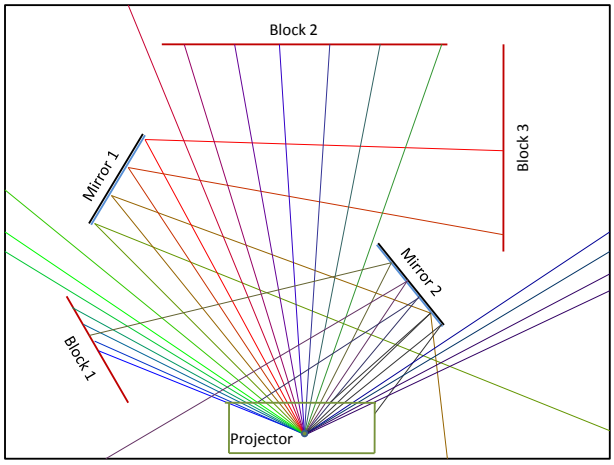

Figure 1: An example of Mirror World with two Mirrors, three Blocks and a Projector.

Your program should take the specifications of the blocks mirrors and projector and draw the path of each light ray in the world as it travels from the projector, bouncing off any mirrors till it either strikes a block or exits the world.

Now enable two different ways of changing the objects in the world and redraw the resulting light field in the 2D world with each step of motion: a) Provide keyboard/mouse based controls to select, rotate and translate each object in the world, and b) Move the objects in a random, but smooth manner.

2.1 Objects in the World

The following are the properties of each element in the world.

1. The World: The world is rectangular and is specified by its size: $width, height$. The origin is supposed to coincide with the center of the world, $width/2, height/2$. The rectangular box around the world may be thought of as four blocks, where the block is as defined below. The world does not move once it is created, while all other object listed below may be moved by the user.

2. Block: A block is an opaque object in the 2D world. Any light ray that falls on either side of the block is absorbed and nothing comes out. The block is line segment and is specified by its end points, (x_1, y_1) and (x_2, y_2).

3. Mirror: A mirror is also a line segment like a block and is specified in the same way: (x_1, y_1) and (x_2, y_2). However, unlike a block, one side of the mirror is shiny and is perfectly reflective, while the other side is perfectly opaque. Any light ray that falls

on the shiny side of the mirror is reflected back as per laws of reflection. The non-shiny side of the mirror acts like a block. The reflective side of the mirror is to the right of the vector pointed from (x_1, y_1) to (x_2, y_2).

4. Projector: A projector is specified using a line segment ((x_1, y_1) and (x_2, y_2)) with n equidistant holes (or pixels) that span the line segment. The point light source is at a perpendicular distance d from the mid point of the line segment, on the right side of the vector pointed from (x_1, y_1) to (x_2, y_2). The light rays are obtained by connecting the point light source to each of the pixels on the line segment and extending the ray further into the world (see Fig. 2). The color of the light ray changes from pure red to pure green to pure blue along the line segment. The projector is defined by the following six numbers: The line segment ((x_1, y_1) and (x_2, y_2)), distance d to the point source, and the number of light rays, n.

2.2 Specifying the World

When the program starts it should be able to take a file (filename taken as command line argument) containing the descriptions of the world. It will contain: 1) The dimensions of the world, 2) Number and specifications of projectors, 3) Number and specifications of the blocks, and 4) Number and specifications of the mirrors. The format of the specifications file is given in Figure 2. Ignore all blank lines in the file. Treat any characters after a # symbol till the end of the line as comment.

3 Controls and Animations

You should be able to select any object by clicking on (or near) it. If there are multiple objects near a click, select any one of the nearby objects. If you find it difficult to implement mouse click based selection (in fact it could be tricky), use the c key to select an object. Repeatedly clicking c should change the selection through all the objects. Highlight any object that is selected (your are free to choose how the highlight is done) and un-highlight it when another object is selected or no objects are selected (by clicking away from any object). One should be able to move the selected object and rotate it about the first point of the line segment. Use the four arrow keys for translation of the object and r and l keys for rotating in clockwise and anti-clockwise directions. Pressing the f and s keys should make any subsequent translation/rotation, faster or slower. Escape or q button is used to quit the interface. Before quitting, your program should save the current state of the world into a file in the same format as in Figure 2.

3.1 Modes of Operation

The world operated in four modes:

1. **Static Mode:** This is the mode described above, where the light rays and objects are static unless the objects are selected and moved by the user as per the controls we defined. The world is in this mode when the program starts and this is the default mode (i.e., when no other mode is explicitly enabled).

2. **Random Walk Mode:** In this mode, each object in the world moves in a smooth, but random manner (say like a drunken walk). For each step of the motion, the world with its light rays are recomputed and drawn. Use the z key enter and exit the random walk mode.

3. **Light Transport Mode:** In this mode, each light ray is drawn as a series of smaller segments, which continuously emanates from the projector and moves along the light path with any reflections in between. Adjust the length of the light segments and the distance between them to obtain a visually pleasing animation, while conveying the information about light reflection and motion. Keep the speed of motion slow; say around $1/4$th of the world width per second. Use the t key to enter and exit the light transport mode.

4. **Gaze at Cursor Mode:** In this mode, the projector will always be pointed at the cursor. You can move the cursor around in the world and the projector will throw light in the direction of the cursor with appropriate reflections and blocks. Use the g key to enter and exit the gaze at cursor mode.

Any of the above action/keypress/motion should result in updating the world, computing the resulting light rays and refreshing the display.

```
# All numbers are in floating point except for number of items
100.0 120.0              # width (x) and height (y) of the world
1                        # number of Projectors
-10.0 -40.0 10.0 -40.0   # Project 1 line: x1 y1 x2 y2
5.0                      # Distance(d) to the point light source
21                       # Number of pixels in the Projector
3                        # number of Blocks
-30.0 -30.0 -30.0 30.0   # Block 1: x1 y1 x2 y2
-25.0 50.0 25.0 50.0     # Block 2: x1 y1 x2 y2
30.0 40.0 45.0 10.0      # Block 3: x1 y1 x2 y2
2                        # number of Mirrors
-20.0 10.0 -10.0 20.0    # Mirror 1: x1 y1 x2 y2
20.0 35.0 20.0 20.0      # Mirror 1: x1 y1 x2 y2
```

Figure 2: Example of a specification file. The world is similar to, but not the same as the example in Fig. 1.

3.2 Optional

1. Add options to insert or delete new blocks mirrors and projectors. Use the$b$, m, and p keys to insert a block, mirror, or projector respectively at a standard position. The inserted object should be highlighted so that it can be readily moved afterwards. Use the delete key to delete a selected object.

2. Provide the option to select multiple modes simultaneously. For example, pressing t and g keys would point the projector at the cursor as it moves, while showing the light transport animation.

3. A Game Mode: Create a game with insects (circles) appearing randomly in the world. The size of the insect shrinks when you shine light from the projector on to the insect, and it vanishes when the size shrinks below a limit. If no light is shining on the insect, it grows in size. You lose a life when the insect grows above a size, and then vanishes. Your goal is to keep your world insect free for as long as possible by moving the position and direction of the projector. The light may fall on the insect either directly or after reflection from a mirror. Think of how you can handle the situation when an insect is between two light rays. Use the p key to enter and exit the $play\ game$ mode.

   You can make the game more interesting (higher levels) by increasing the rate at which insects appear, changing the angle of light rays that are coming out from the projector, introducing more blocks in the world, introducing motion of the insects, etc.

4 Useful Hints

1. Write a few basic functions for the following and use them in drawing/animating your world. The first is a $drawLine$ function to draw a line segment of specified end points, thickness and color. Use this to define draw functions for each of the objects in the world. Finally, define a $drawWorld$ function that would draw the world according to the current state of the world.

2. Other useful functions would include: $findIntersection$ that would find the intersection of a line with objects in the world and return the closest object or line segment, the intersection point; functions for translation and rotation of each object; and insertion and deletion of objects. If your program is object oriented, the translation and rotation could be functions of the object class.

3. To do animation and games, use a regular timed function that changes the state of the world when the clock passes specific ticks (say every 10th of a second). In any case you should decouple the speed of drawing and the update steps of the world.

5 Submission

You submissions should include your source code, a makefile and a readme file that explains the compilation and execution process. Also include any additional information that is needed to evaluate your submission including a description of the optional items in the readme file. Do not use any libraries other that gl, glu, glut and standard C, C++ libraries.

Details of how to submit and any modification to the above submission details will be posted by the TAs towards the submission deadline.

6 Grading

You will be graded based on the correctness and efficiency (speed) of the implementation of the minimum elements described above. This will contribute to 90% of your grade. Remaining 10% will be given based on the improvements that you do over the basic game. In addition, submissions that are found to be exceptional by the graders, will be showcased, and will be awarded extra credits up to 10%.