COSC363 Computer Graphics
Lab03: Illumination

Aim:

This lab provides an introduction to the OpenGL illumination model, different forms of lighting and the associated lighting effects (shadows, reflections) that could be used for rendering a scene.

I. A toy train:

1. The program **`train.cpp`** displays a scene consisting of circular rail tracks, and the model of a toy locomotive (Fig. (1)). A description of the functions included in the program is given below.

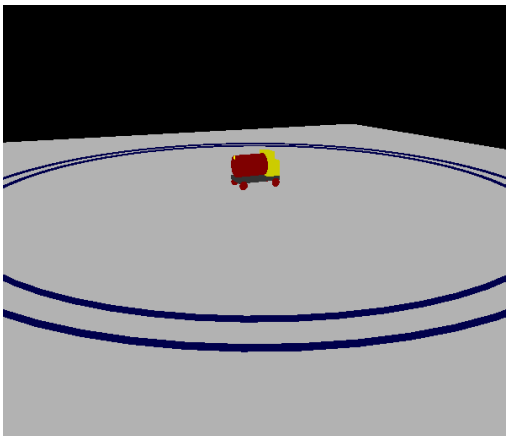

Fig. (1)

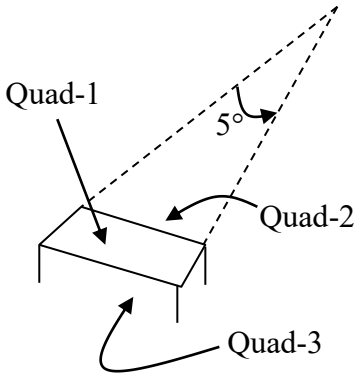

Fig. (2)

`floor()`: The floor plane is *not* modeled as a single quad. Instead, it is an array of 200x200 tiny quads laid out on the *xz*-plane. This form of subdivision of the floor plane is useful for proper lighting of the scene.

`track()`: This function creates the model of a circular rail track of specified radius using 72 small segments, each segment subtending 5 degs at the centre (Fig (2)). Each segment is modeled as a set of 3 quads. The tracks have a height of 1 unit, and a separation of 10 units (Fig. (3)).

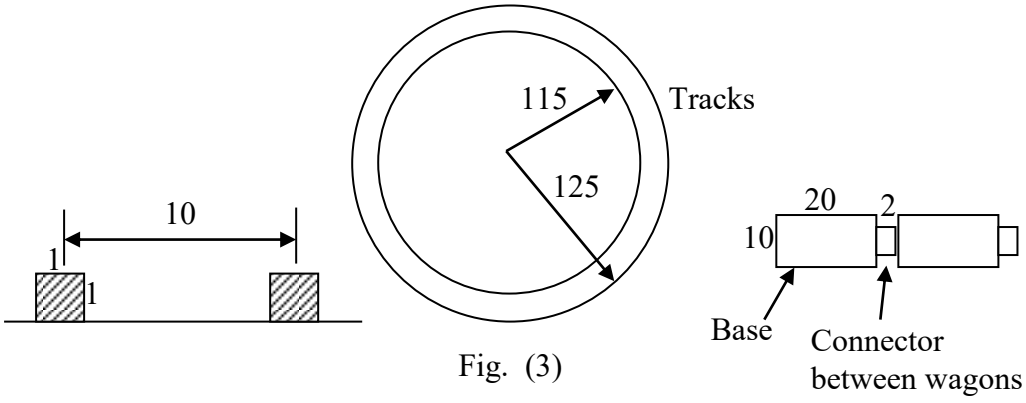

Fig. (3)

`base()`: The locomotive and the wagons of the train have a common base to which 4 wheels are attached (See Figs. (3)-(5)). Each base has a size 20x10 units. A connector of size 2 units is also attached to the base.

`engine()`: This function creates the model of the locomotive including the base, a boiler and the cab (Fig. (4)).

`wagon()`: This function models a wagon as a combination of the base and a cube (Fig. (5)).

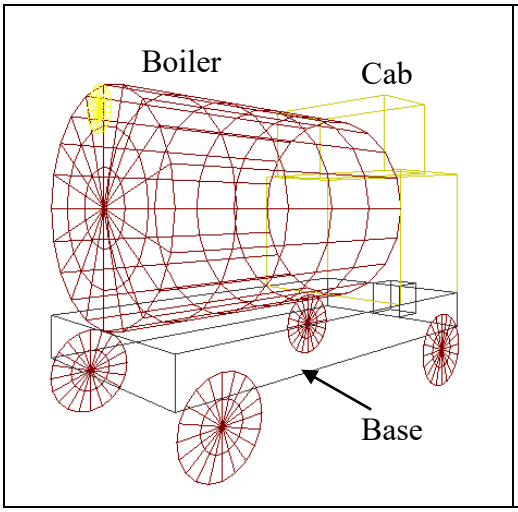

Fig. (4)

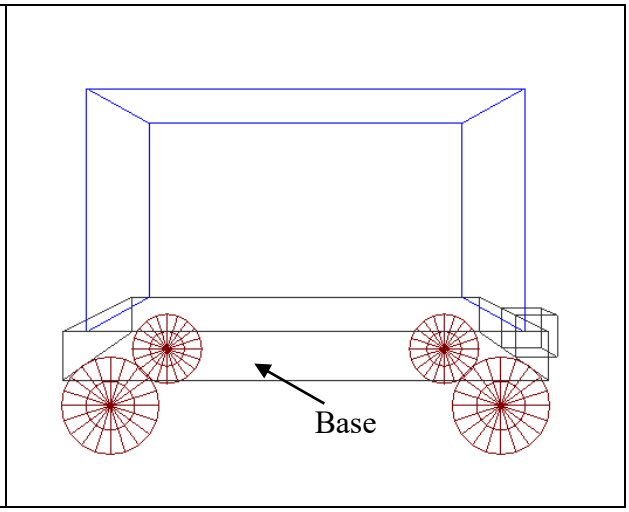

Fig. (5)

2. Lighting is currently disabled. Surfaces are rendered using only constant colour values specified by the glColor3f() function, and therefore appear flat. Enable lighting by un-commenting the statement glEnable(GL_LIGHTING) in the initialize() function. The scene is now illuminated with a light source GL_LIGHT0 positioned at (0, 50, 0) directly above the origin. However, OpenGL has ignored all surface colour values that were defined using glColor3f() (Fig. (6)).

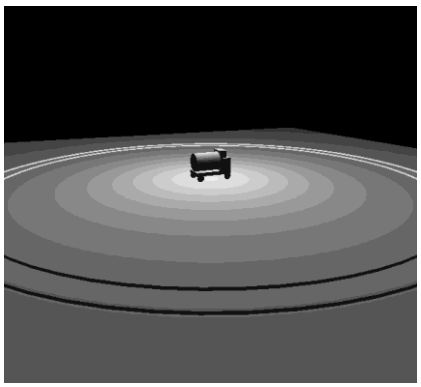

Fig. (6)

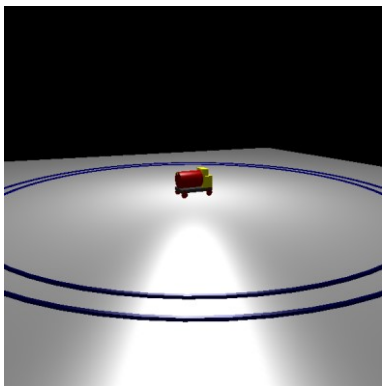

Fig. (7)

3. When lighting is enabled, OpenGL expects the material properties for each surface to be defined using three calls to the glMaterial3fv() function with ambient, diffuse and specular colour values. The ambient and diffuse colours are usually the surface colour already defined using glColor3f(). OpenGL can

use this colour value, instead of ignoring it, as the ambient and diffuse properties if the following state is enabled:

```
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
```

The common specular colour and the shininess term for all materials need be set only once:

```
glMaterialfv(GL_FRONT, GL_SPECULAR, white);
glMaterialf(GL_FRONT, GL_SHININESS, 50);
```

Include the above statements in initialize(), and the display should change to that given in Fig. (7). Notice the bright specular highlight on the floor plane.

4. Floors are generally diffuse surfaces. We can suppress specular reflections from the floor by temporarily setting the specular colour to black. Include the following statement in the `floor()` function before the glBegin() statement:
   ```
   glMaterialfv(GL_FRONT, GL_SPECULAR, black);
   ```
   Reset the specular property to white after the glEnd() statement:
   ```
   glMaterialfv(GL_FRONT, GL_SPECULAR, white);
   ```
   The floor plane should now be displayed as a diffusely reflecting surface (Fig. 8)

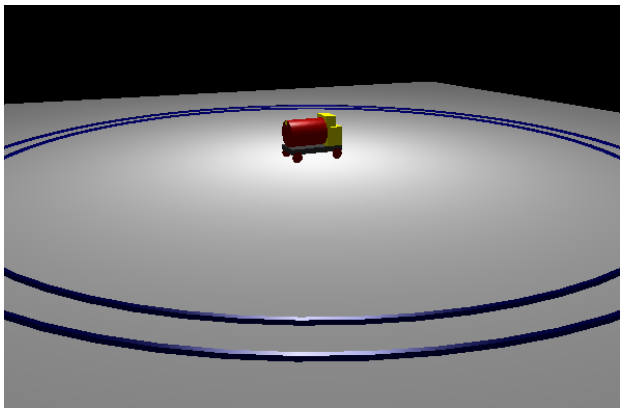

Fig. (8)

5. We will now implement a series of transformations to construct a toy train and to make it move along the tracks! Move the rail engine from its current position at the origin to the position (0, 1, –120). The *y*-value 1 corresponds to the track height, and the *z* value 120 corresponds to the mean radius of the tracks (see Fig. (9)). Please make sure to include the transformation within a `glpushMatrix()-glPopMatrix()` block so that it will not affect other objects in the scene.

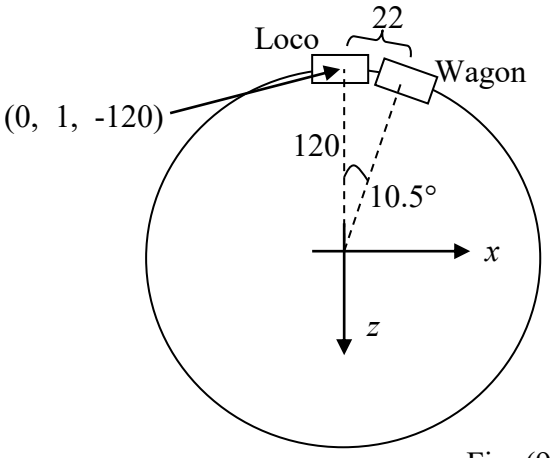

Fig. (9)

6. Generate the display of a wagon by calling the function wagon(). Translate it from the origin to (0, 1, −120) and then rotate it about the *y*-axis by 10.5 degrees (positive or negative?) to position it correctly as shown in Fig (9). (How was the separation angle 10.5 degs obtained?). Add a few more wagons to the train (Fig.(10)). Use a timer call back to move the entire train along the track by applying a continuous rotation about the *y*-axis. Adjust the timer delay and angle step size to get a smooth animation. Note that the program uses double-buffered animation to eliminate screen flicker. This is done by specifying GLUT_DOUBLE in `glutInitDisplayMode()` function, and by calling `glutSwapBuffers()` after drawing the scene.

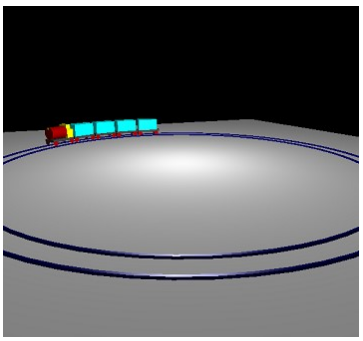

Fig. (10)

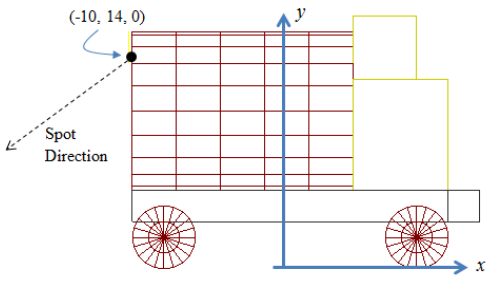

Fig. (11)

7. Create a new light source GL_LIGHT1 with the same parameters as GL_LIGHT0. Convert it to a spotlight with cutoff angle 30 degs by adding the following statements (see slide[3]-18):

```
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 30.0);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT,0.01);
```

Inside `display()` function, specify the spotlight's position as (-10, 14, 0). This corresponds to the headlight position on the locomotive (Fig (11)). Also specify the direction of the spotlight such that it is oriented towards the floor. All transformations applied to the locomotive must also be applied to the spotlight's position and direction, so that it moves as if attached to the engine (Fig. (12)).

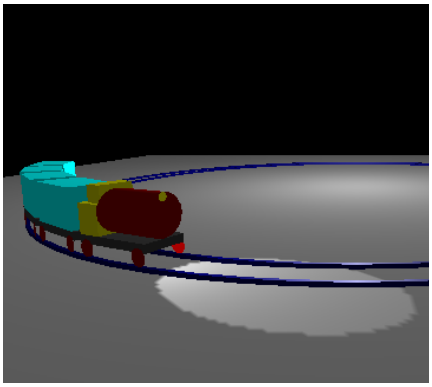

Fig. (12)

8. Move the camera backwards (towards the viewer) by changing its *z*-position in gluLookAt() from 180 to 250.

II. Shadow and reflection:

The program **`Shadow.cpp`** displays a floor plane and a rotating teapot. The floor plane contains a small rectangular section to which a glass texture is mapped (Fig. (13)). Please ignore the code for loading and mapping textures. The light is positioned towards the right side of the scene, with coordinates (80, 80, 0).

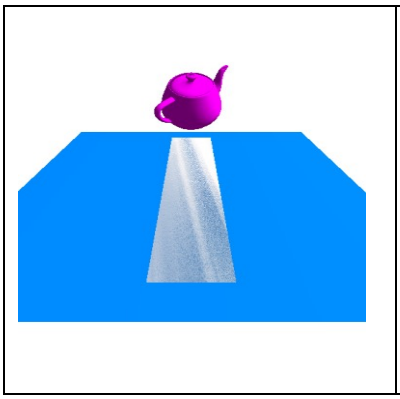

Fig. (13)

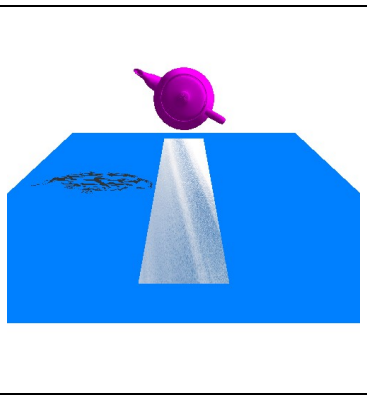

Fig. (14)

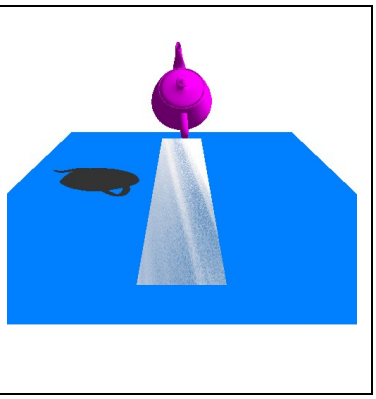

Fig. (15)

1. Inside the `display()` function, define a shadow matrix as given on slide [7]-25, and draw the teapot again (along with all its transformations), but this time also multiplying by the shadow matrix. Remember to change the colour of the teapot's shadow to a dark gray value, and disable lighting so that the shadow polygons will have uniform colour. Enable lighting again after drawing the shadow. The shadow is rendered on the floor plane (with y-value 0) and therefore results in an artefact called depth fighting (or *z*-fighting) because of equal depth values for fragments belonging to the floor and the shadow (Fig. (14)). Move the floor plane slightly downward, by setting the value of the variable `floor_height` (in function floor()) to –0.1 The shadow of the teapot should now get rendered correctly (Fig. (15)).

2. We will now render the reflection of the teapot on the glass plate. Draw the teapot (along with all its transformations) for the third time, but this time also multiplying by the scale transformation glScalef(1, -1, 1) which inverts the teapot along the *y*-axis. Note: The teapots must be drawn first, and finally the

floor plane. The reflection of the teapot will not be visible as it is occluded by the floor plane.

3. Set the transparency value of glass (in function floor()) to a small value, say 0.3. This is the fourth parameter in `glColor4f(…)` called before drawing the glass plate. Note that blending is already enabled in the program. Please make sure that the texture environment parameter is set to GL_MODULATE. The output of the program is shown in Fig. (16).

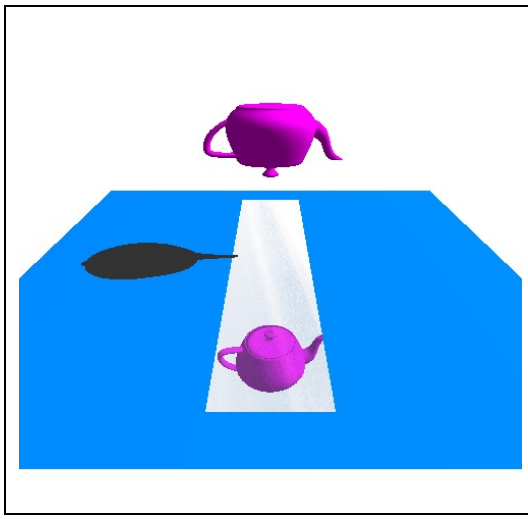

Fig. (16). Incorrect reflection.

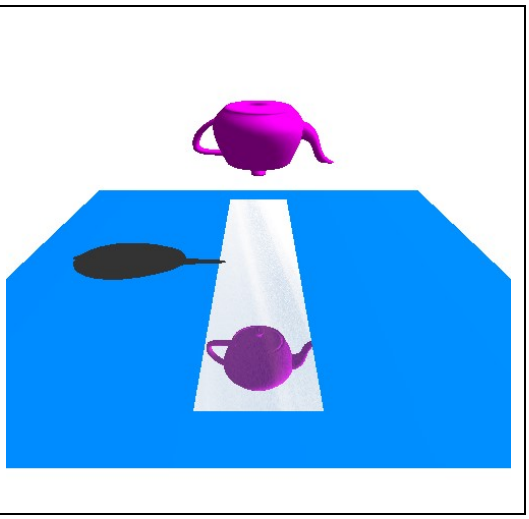

Fig. (17). Correct reflection

4. The reflection in Fig. (16) is not rendered correctly. The lid of the teapot is in shadow, but the lid of its reflection is illuminated by the light source. In fact, the whole reflected object is illuminated from the light source's position. We need to invert the light source's *y*-value before drawing the reflection:

```
//  Invert light's position before drawing reflection
      light[1] = -light[1];
      glLightfv(GL_LIGHT0, GL_POSITION, light);   //new position
//  Draw Reflection
     ...
```

5. Change the light's position back to its original value after drawing the reflected teapot (and before drawing the floor). The final output is in Fig. (17).

III. Quiz-03

The quiz will remain open until **5pm, 22-Mar-2019**.

A quiz can be attempted only once. A question within a quiz may be attempted multiple times. However, a fraction of the marks (25%) will be deducted for each incorrect answer.

[3] COSC363 Lecture Slides "3 Illumination".