COSC363 Computer Graphics
Lab03: Illumination and Texture Mapping

Aim:

This lab provides an introduction to the OpenGL illumination model and texture mapping functions.

I. Train.cpp:

1. The program **`train.cpp`** displays a scene consisting of circular rail tracks, and the model of a toy locomotive (Fig. (1)). This model serves as a good example of a 3D object constructed using GLUT objects. A description of the functions included in the program is given below.

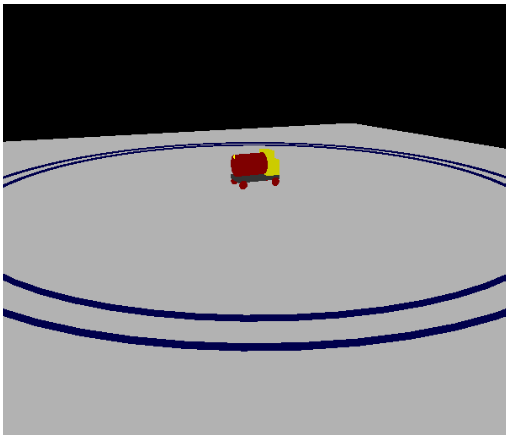


Fig. (1)

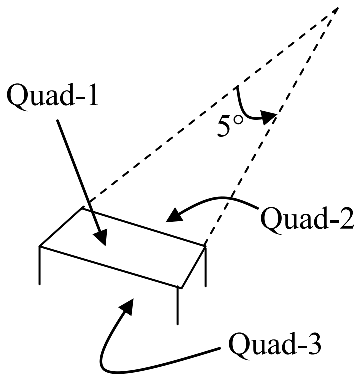


Fig. (2)

`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)).

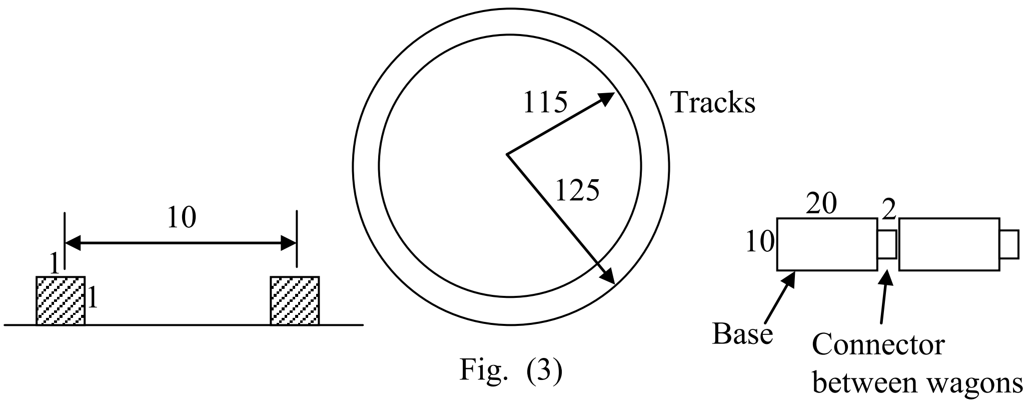


Fig. (3)

`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)).

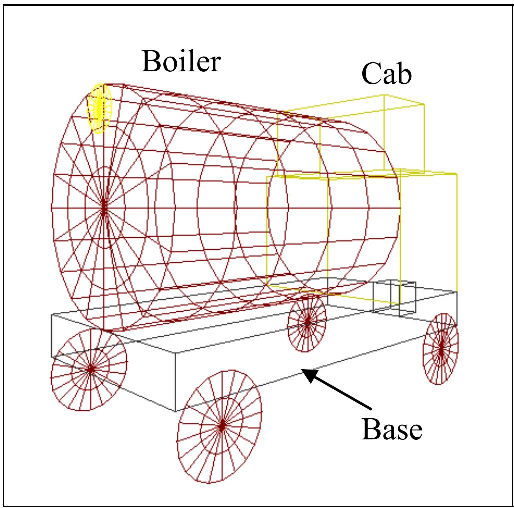


Fig. (4)

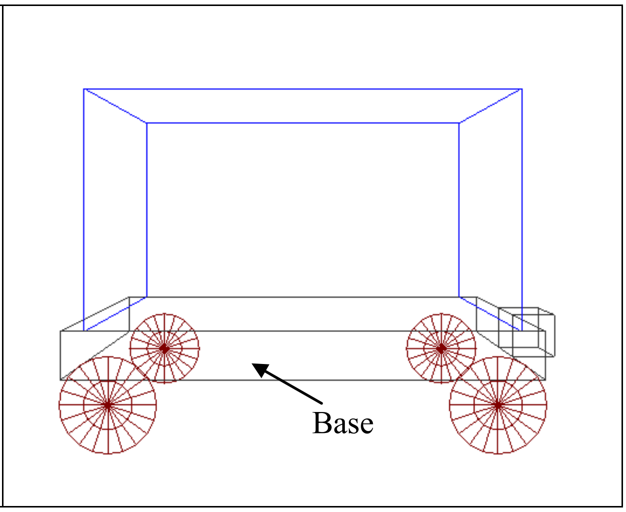


Fig. (5)

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)).

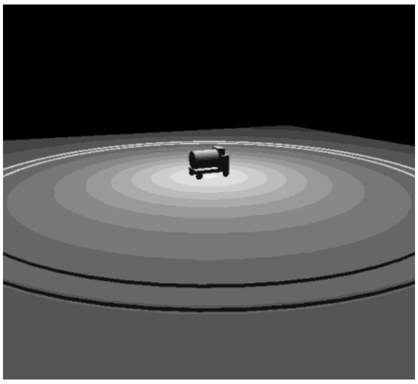


Fig. (6)

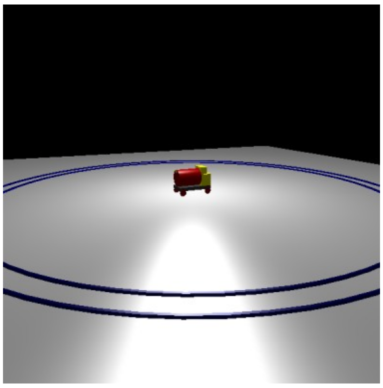


Fig. (7)

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)

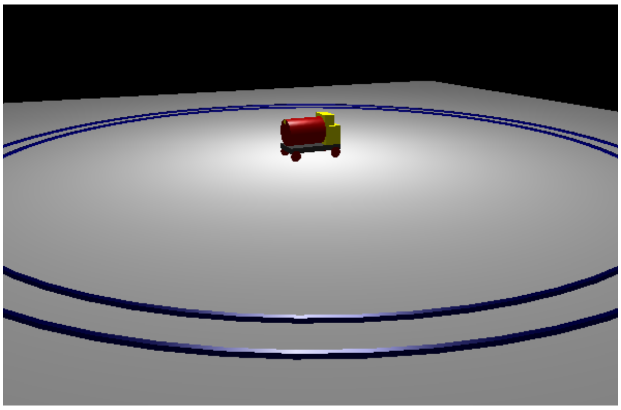


Fig. (8)

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.

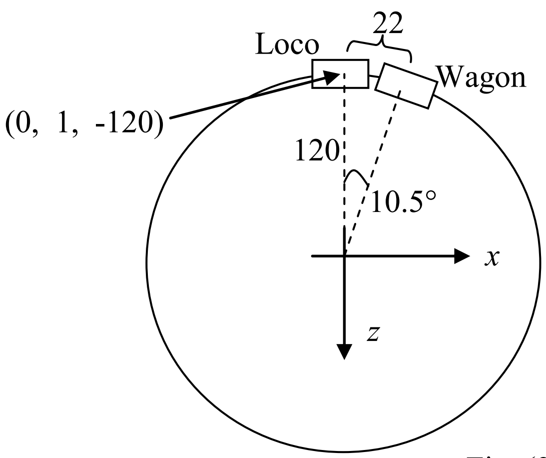


Fig. (9)

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.

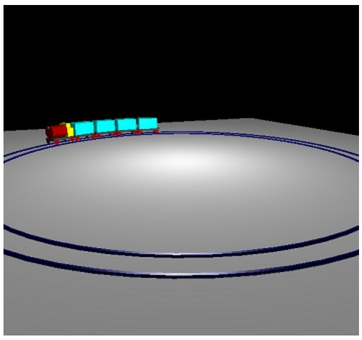


Fig. (10)

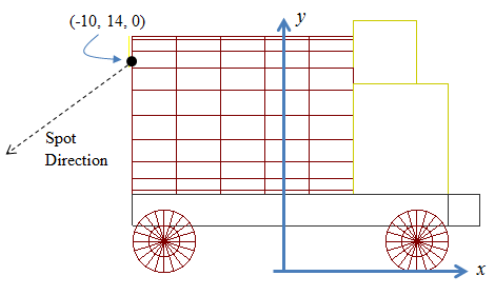


Fig. (11)

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]-14):

```
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)).

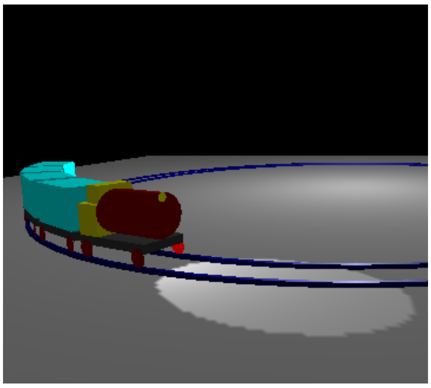


Fig. (12)

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

Congratulations! You have just created a three-dimensional animated model of a cool toy train.

II. RailWagon.cpp:

The program displays the model of a rail wagon (Fig. (a)). The arrow keys can be used to rotate the model and to move the camera up or down. The body of the wagon is generated using 5 quads to facilitate texture mapping (Fig. (b)). It consists of three large sides “Front”, “Top” and "Back", and two smaller sides “Left", and "Right” (see function `wagon()`).

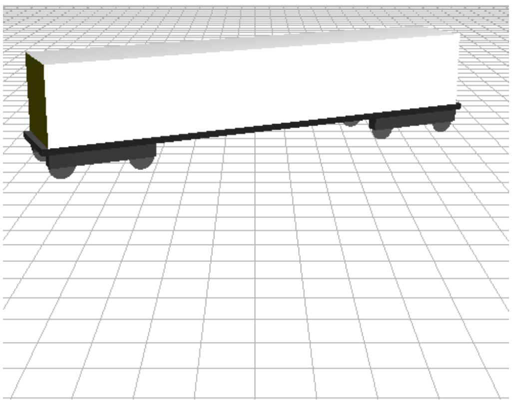


Fig. (a)

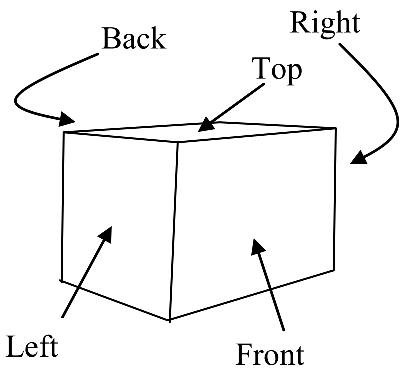


Fig. (b)

1. Uncomment the line marked “<<<<” inside the `initialize()` function. This statement calls the function `loadTexture()`. The function is defined at the beginning of the program. It loads an image in bitmap format, "WagonTexture.bmp", of size 512x256 pixels, and sets the texture parameters.

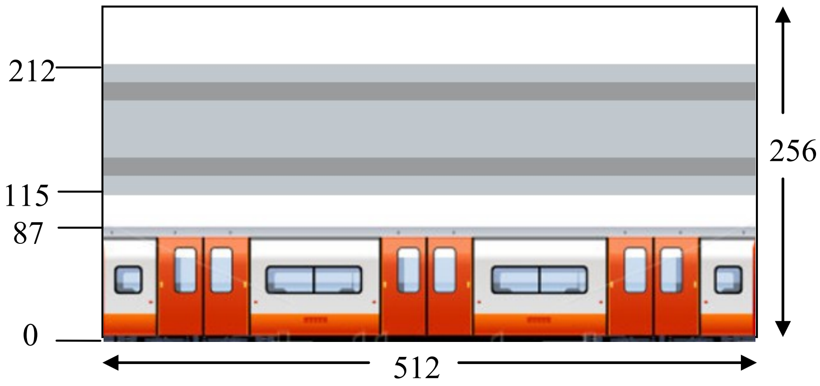


Fig. (c)

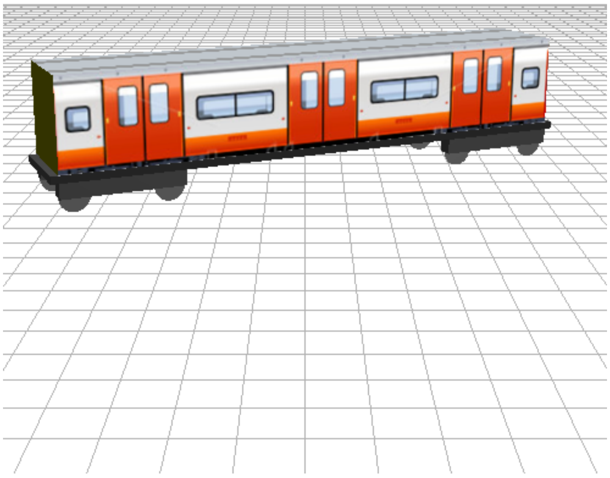
2. We will map textures to the three main sides of the wagon only. The texture image consists of two sections. The bottom section (pixel rows 0 to 87) will be mapped to the front and back sides of the wagon and the top section (pixel rows 115 to 212) to the top side. Using the values given in Fig.(c), compute the texture coordinates that must be assigned to the vertices of the three quads.

3. Inside the `wagon()` function, assign texture coordinates to every vertex of the three quads that define the main sides of the wagon. Also enable texturing of the three quads as shown in Fig. (d)

```
void wagon()
{
        base();

        glColor4f(1.0, 1.0, 1.0, 1.0);

        glEnable(GL_TEXTURE_2D);
     →  glBindTexture(GL_TEXTURE_2D, txId);

        //3 large polygons (front, back, top)
        glBegin(GL_QUADS);
          glNormal3f(0.0, 0.0, 1.0);    //Facing +z (Front side)
     →    glTexCoord2f(0., 0.);     glVertex3f(-35.0, 5.0, 6.0);
     →    glTexCoord2f(1., 0.);     glVertex3f(35.0, 5.0, 6.0);
     →    glTexCoord2f(         );    glVertex3f(35.0, 17.0, 6.0);
     →    glTexCoord2f(         );    glVertex3f(-35.0, 17.0, 6.0);
 •
 •
```

Fig. (d)

4. Disable texturing (using `glDisable(GL_TEXTURE_2D)`) of the two small sides of the wagon, the base and the floor. The program should produce the output shown in Fig. (e).



. Fig. (e)

5. Change the texture environment parameter from GL_REPLACE to GL_MODULATE. What effect does this change produce?

III. Yard.cpp:

1. This section uses texture images in TARGA (.tga) format. The program displays five polygons (four "walls" and one "floor") forming a rectangular yard (Fig. (f)). The camera is placed at the center of the yard. The arrow keys can be used to change the view direction and to move in the current direction.

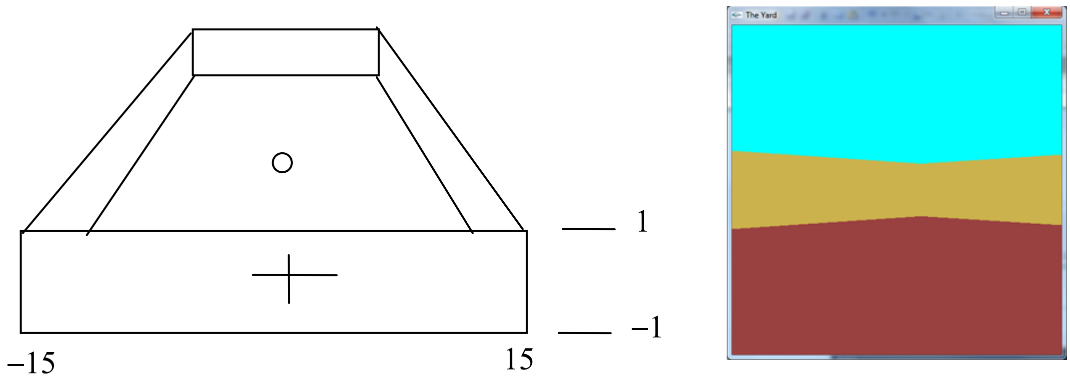
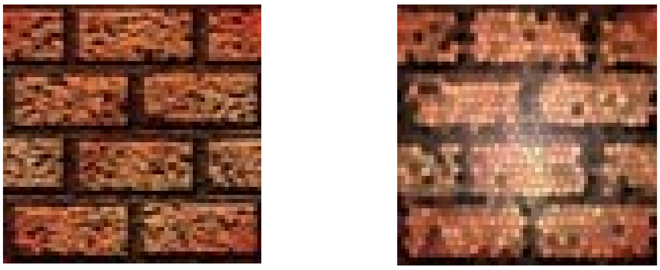


Fig. (f)

2. Two textures "Wall.tga" and "Floor.tga" are provided (Fig. (g)). The textures have a special property that they can be seamlessly tiled any number of times along both horizontal and vertical (s, t) directions without any visible discontinuities at boundary pixels. The program contains the necessary function definition to load both these textures. Note also that GL_REPEAT is the default wrapping mode for textures. Load the two textures and enable texture mapping by un-commenting the first two statements inside the

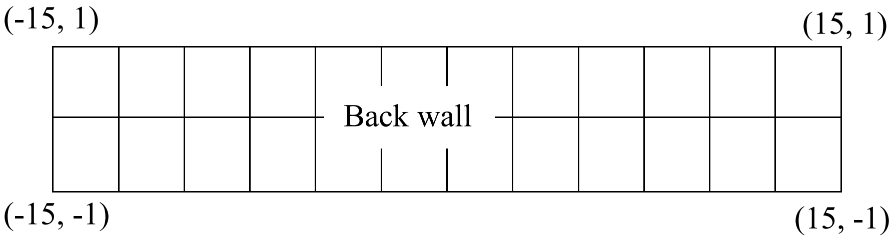
`initialise()` function. The program uses the header file **`loadTGA.h`** to read tga files.



Wall.tga Floor.tga

Fig. (g)

3. Using the image "Wall.tga", texture the first quad (in the function wall()) as given below. The texture must be repeated 12 times in the horizontal direction, and twice along the vertical direction.




```
glTexCoord2f(0.0,  2.0);      glVertex3f(-15, 1, -15);
glTexCoord2f(0.0,  0.0);      glVertex3f(-15, -1, -15);
glTexCoord2f(12.0, 0.0);      glVertex3f(15, -1, -15);
glTexCoord2f(12.0, 2.0);      glVertex3f(15, 1, -15);
```

Texture the remaining 3 quads also the same way.

4. Similarly, texture map the floor using the second texture image "Floor.tga", with a repetition count of 16 along both directions. The final output is shown in Fig. (h).

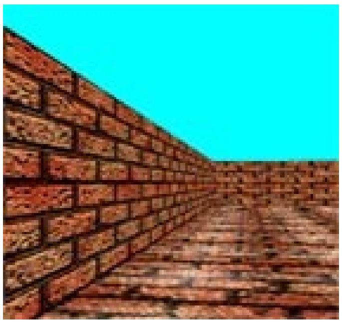


Fig. (h)

5. The program implements an "interactive walk-through" mode of the camera, where the up (down) arrow key moves the camera forward (backward), and the left and right arrow keys change the direction of movement.
   Take-home exercise: Implement a simple collision detection algorithm so that the camera would always remain within the four walls of the yard.

[3] COSC363 Lecture Slides "3 Illumination".

IV. Quiz-03

The quiz will remain open until **5pm, 20-Mar-2020**.

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.