COSC363 Computer Graphics
Lab04: Texture Mapping

Aim:

This lab provides an introduction to OpenGL functions used for mapping textures to object models.

I. 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()`).

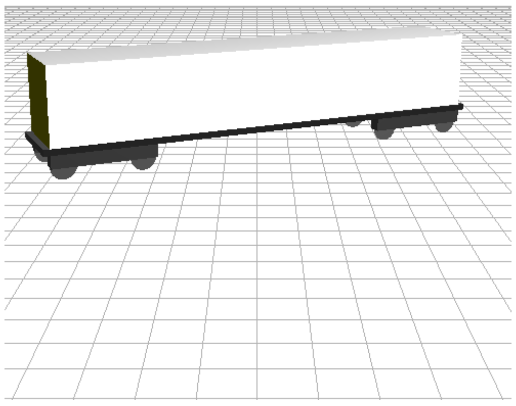


Fig. (a)

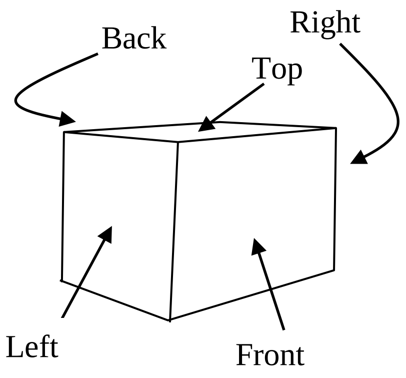


Fig. (b)

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.

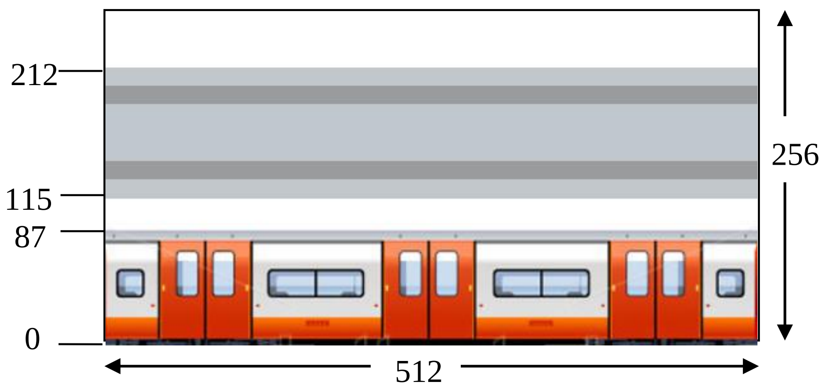


Fig. (c)

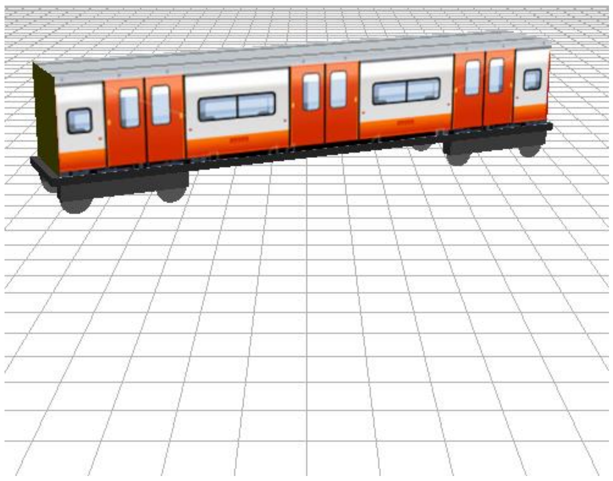
2. We will map textures only to the three large sides (front, back, top) of the wagon. The texture image consists of two sections. The bottom section (pixel rows 0 to 87) will be mapped to the front and back sides of the wagon and the top section (pixel rows 115 to 212) to the top side. Using the values 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).

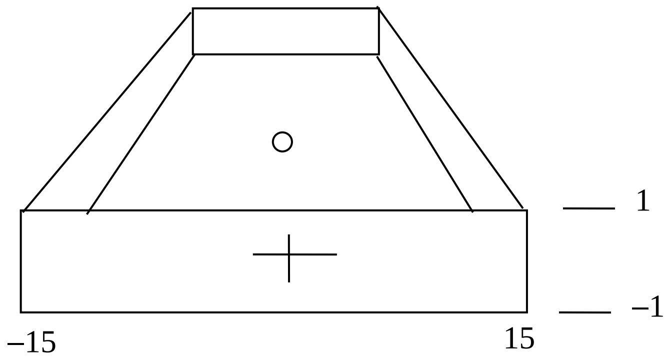


. Fig. (e)

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

II. 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.



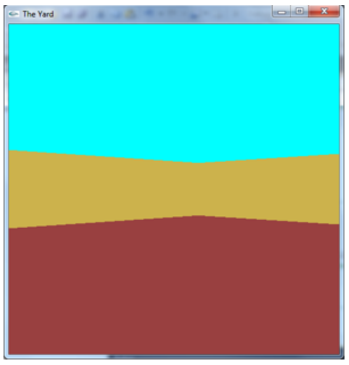


Fig. (f)

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

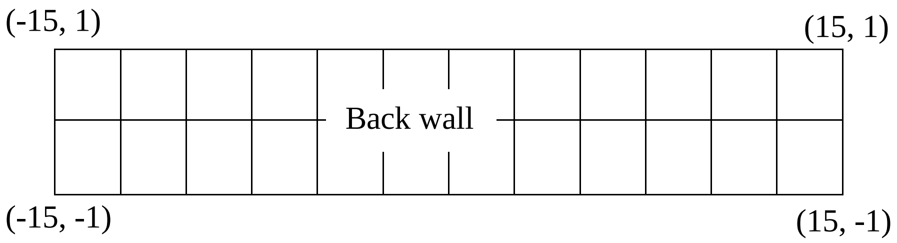
`initialise()` function. The program uses the header file **`loadTGA.h`** to read tga files.

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).

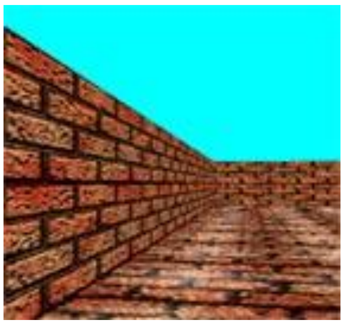


Fig. (h)

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.

III. MapleLeaf.cpp

1. Last week, you developed a program to display the object model shown in Fig.1(a). We will now map a texture (Fig. 1(b)) onto this model. Please add the statements
   `#include "loadBMP.h"`
   and `GLuint txId;`
   at the beginning of your program and also copy in the `loadTexture()` function from RailWagon.cpp. Please change the image file name to `WoodTexture.bmp` and the texture environment parameter defined in this function from GL_REPLACE to GL_MODULATE. Inside the `initialise()` function, add the statements `loadTexture();` and `glBindTexture(GL_TEXTURE_2D, txId);`.

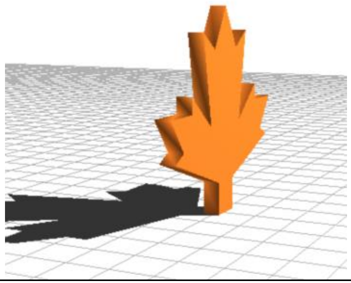
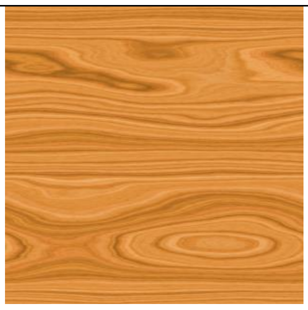
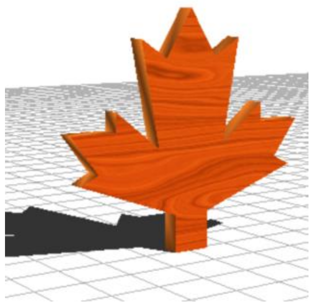
		

Fig. 1(a)	Fig. 1(b)	Fig. 1(c)

2. Please modify the function `drawTriangulated()` as given below.

```
glEnable(GL_TEXTURE_2D);
 glBegin(GL_TRIANGLE_FAN);
   glTexCoord2f(s(0), t(6));
   glVertex3f(0, 6, 0);
   for (int i = 0; i < N; i++)
   {
      glTexCoord2f(s(vx[i]), t(vy[i]));
      glVertex3f(vx[i], vy[i], 0);
   }
   glTexCoord2f(s(vx[0]), t(vy[0]));
   glVertex3f(vx[0], vy[0], 0);
 glEnd();
glDisable(GL_TEXTURE_2D);
```

3. The computation of texture coordinates in the above method is based on the normalization of *x*, *y* coordinates using the formulae given on slide [5]-20. Please include the corresponding functions `float s(float x)` and

`float t(float y)` given on slide [5]-21, in the program. Use the following values for the min, max values of the model coordinates.

XMIN = –14, XMAX = 14, YMIN = 0, YMAX = 25.

The output of the program is shown in Fig. 1(c).

4. In the output generated by your program, you will notice that the shadow region also contains the mapped texture! Please make the required changes in your program to disable texture mapping when drawing the shadow object.

IV. Vase.cpp (Take home exercise)

Extend the Vase.cpp program developed last week to include texture mapping functions to load and map a bitmap texture "VaseTexture.bmp".
In the `display()` function, enable texture mapping. Please refer to Slide [5]-23 for more information on computing texture coordinates for the vertices of polygonal strips. The textured output is shown in Fig 2.

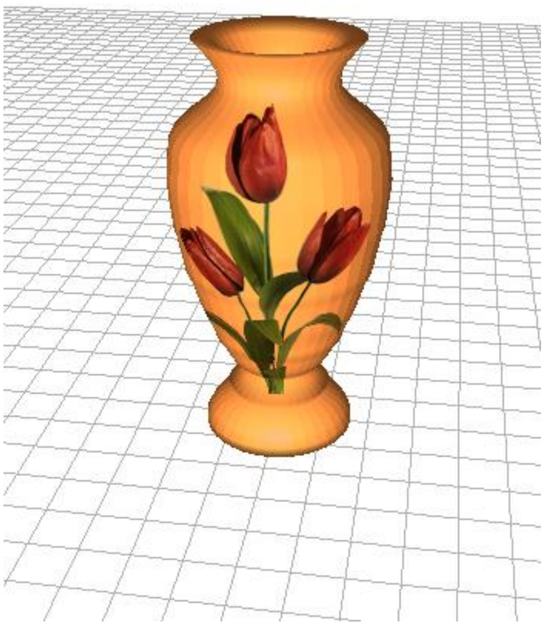


Fig. 2.

V. Quiz-04

The quiz will remain open until **5pm, 7-Apr-2021**.

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.