COSC363 Computer Graphics

Lab04: Texture Mapping

Aim:

This lab introduces methods for mapping textures to polygonal faces of objects. You will learn to load image files in two different formats, and to map them to different sections of a three-dimensional object.

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()`).

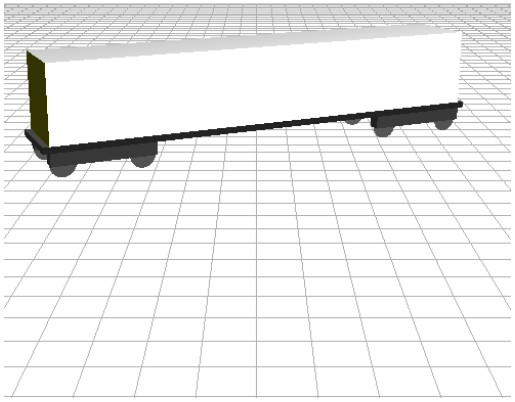


Fig. (a)

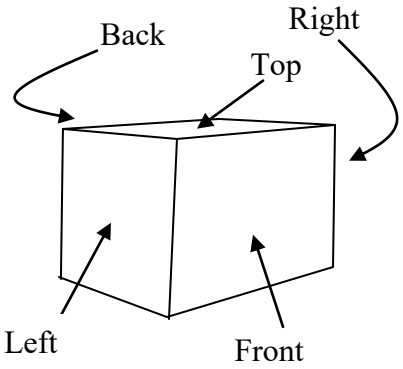


Fig. (b)

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.

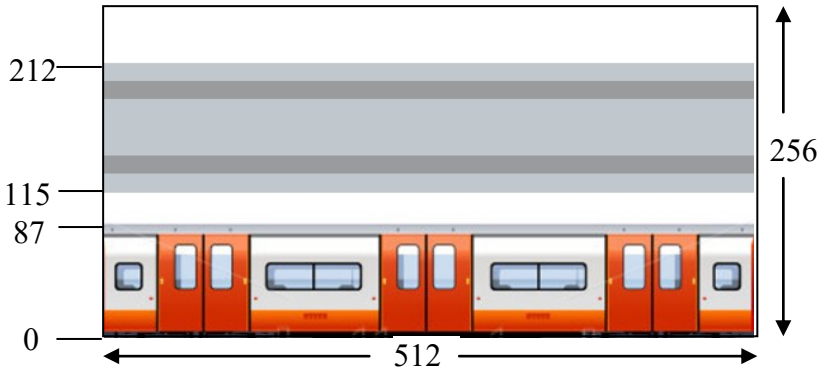


Fig. (c)

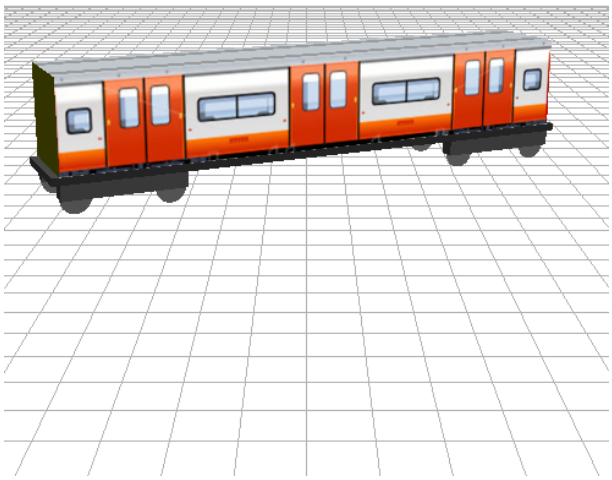
2. We will map textures to the three main sides of the wagon only. The texture image consists of two sections. The bottom section (pixel rows 0 to 87) will be mapped to the front and back sides of the wagon and the top section (pixel rows 115 to 212) to the top side. Using the values given in Fig.(c),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

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.

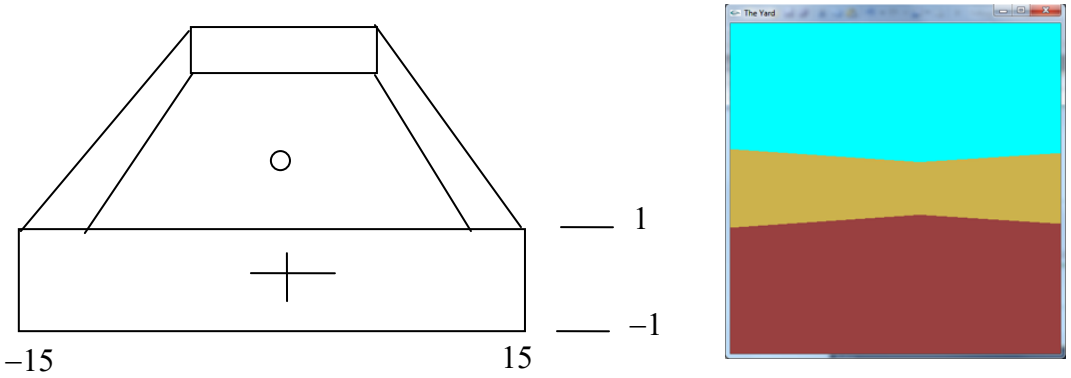
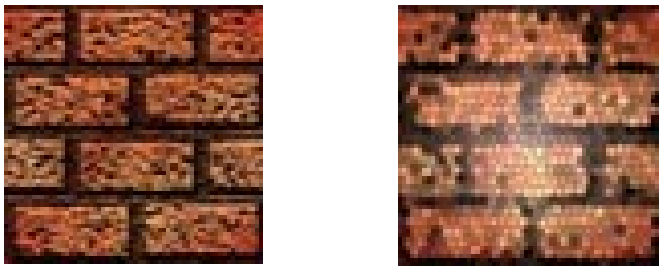


Fig. (f)

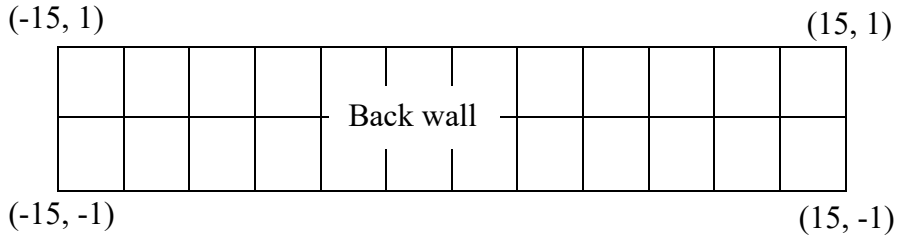
2. Two textures "Wall.tga" and "Floor.tga" are provided (Fig. (g)). The textures have a special property that they can be seamlessly tiled any number of times along both horizontal and vertical (s, t) directions without any visible discontinuities at boundary pixels. The program contains the necessary 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.

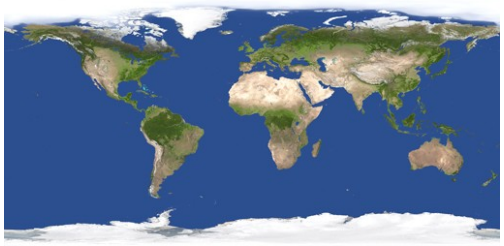
4. Similarly, texture map the floor using the second texture image "Floor.tga", with a repetition count of 16 along both directions. The final output is shown in Fig. (h).

Fig. (h)

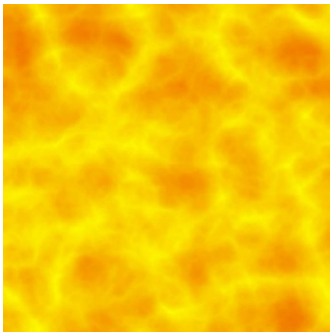
5. The program implements an "interactive walk-through" mode of the camera, where the up (down) arrow key moves the camera forward (backward), and the left and right arrow keys change the direction of movement.
   Take-home exercise: Implement a simple collision detection algorithm so that the camera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

III. Earth.cpp

The program "Earth.cpp", uses the header file `loadBMP.h` to load two texture images in BMP format (Fig (1)). The program gives an example of texturing spheres defined as quadric surfaces using implicitly generated texture coordinates.



Earth.bmp



Sun.bmp

Fig. (1)

There exists a one-to-one mapping of points on a quadric surface and points on a two-dimensional image. Texture coordinates for such surfaces can be automatically generated and assigned to vertex coordinates by specifying the **auto texture mapping** mode for quadric surfaces using the function call

```
gluQuadricTexture (q, GL_TRUE);
```

This mode is already enabled inside the function initialise().

The program generates a display shown in Fig. (2). The sphere representing the Sun is positioned at the origin, and the Earth at a distance of 20 units along the x-axis. Please note how the Earth is first rotated about its axis and then translated to (20, 0).

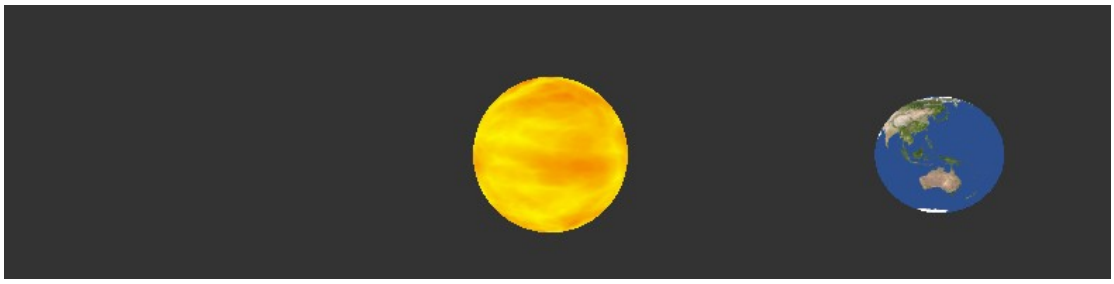


Fig. (2)

Specify a light source at the origin (position of the Sun). Light parameters have already been defined inside the `initialise()` function. Modify the environment parameter of the Earth texture to GL_MODULATE in the display() function, so that shadows are made visible on Earth's textured surface (Fig (3)):

```
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

Finally, revolve the Earth around the Sun (Fig. (3)).

Fig. (3)

IV. (Take-home exercise) Skybox.cpp

A skybox is a large cube-shaped structure placed in a scene to which a set of textures is mapped to generate the appearance of a natural surrounding environment. The camera usually has a fixed position at the centre of the scene, and can only rotate to give different views of the environment. Six textures corresponding to the six sides of a cube as shown in Fig. (4) are provided.

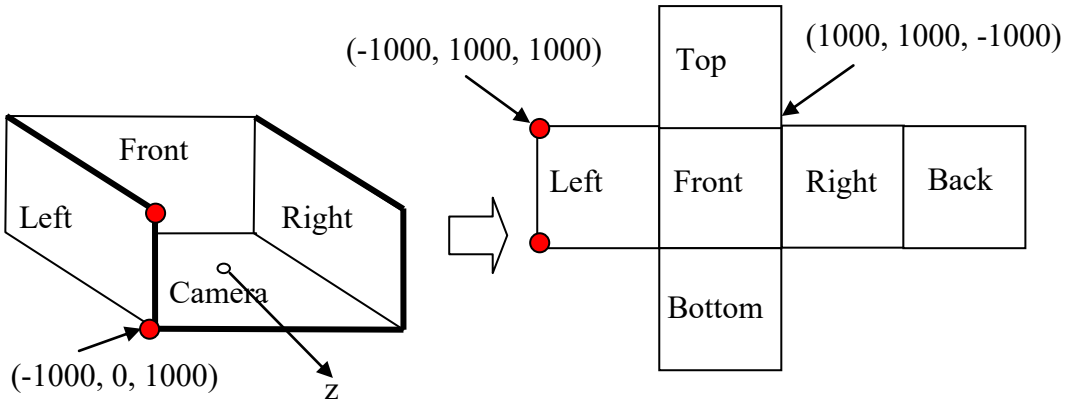
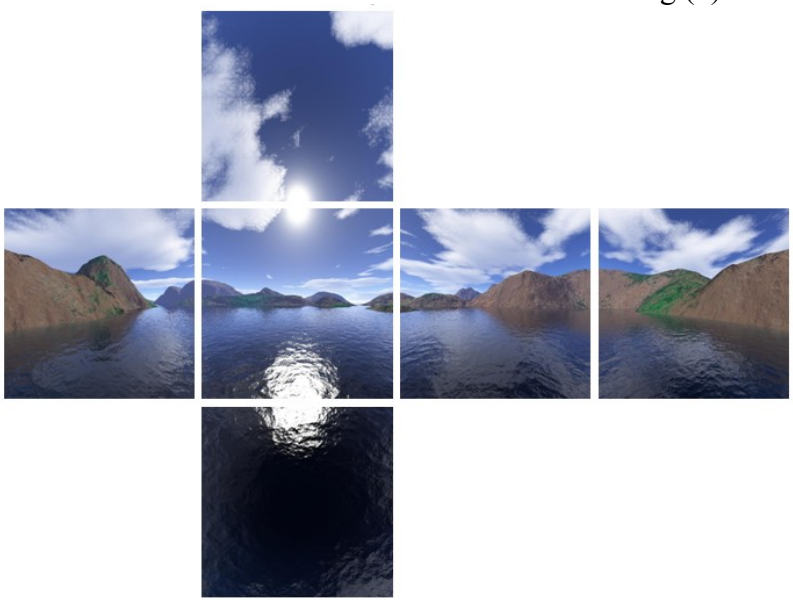


Fig (4)



The program `Skybox.cpp` displays the six sides of a cube with the camera positioned at the origin. The camera can be rotated using left/right arrow keys. The sides of the box are assigned different colours to make them clearly visible (Fig. 5a). The program already includes functions to load the textures and to set up texture parameters. You only need to assign texture coordinates using the function `glTexCoord2f( , )` to each vertex of each of the cubes to generate the display of a skybox as shown below (Fig. (5b)). Please check (by rotating the

camera) if all sides of the cube are properly mapped with seamless tiling and continuity of images across the edges.

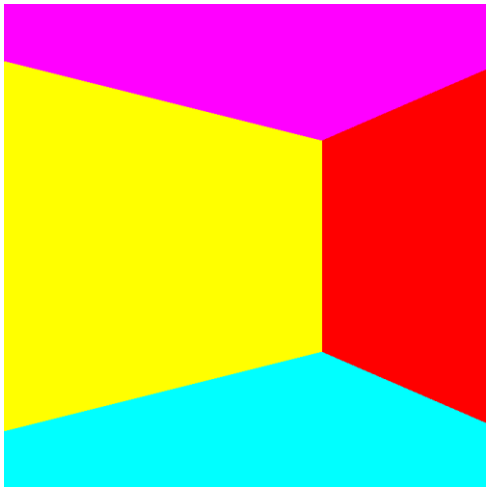


Fig (5a)

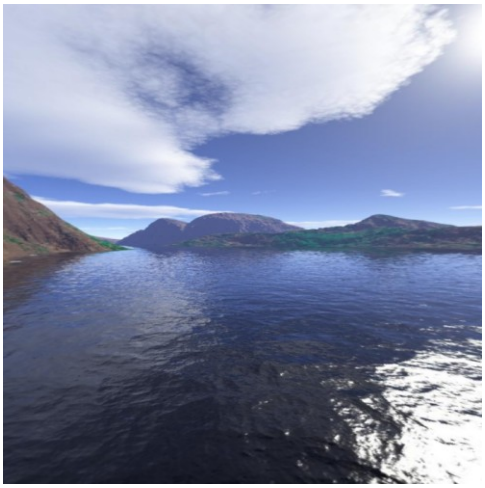


Fig (5b)

NOTE:

Texture images must have the following properties in order to be loaded correctly using the functions defined in loadBMP.h, loadTGA.h:
BMP files: Windows bitmap, 24 bits, Width and height must be a power of 2.
TGA files: Uncompressed TGA, 24 bits, Width and height must be a power of 2.

V. Quiz-04

The quiz will remain open until **5pm, 29-Mar-2019**.