CHAPTER 1

INTRODUCTION

1.1 Introduction to Computer Graphics

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. The field began humbly almost 50 years ago, with the display of a few lines on a cathode-ray tube (CRT); now, we can create images by computer that are indistinguishable from photographs of real objects. We routinely train pilots with simulated airplanes, generating graphical displays of a virtual environment in real time. Feature-length movies made entirely by computer have been successful, both critically and financially. Massive multiplayer games can involve tens of thousands of concurrent participants.

Perhaps the dominant characteristic of this new millennium is how computer and communication technologies have become dominant forces in our lives. Activities as wide – ranging as filmmaking, publishing, banking and education continue to undergo revolutionary changes as these technologies alter the ways in which we conduct our daily activities. The combination of computers, networks, and the complex human visual system, through computer graphics, has led to new ways of displaying information, seeing virtual worlds, and communicating with people and machines.

The Computer Graphics is one of the most effective and commonly used methods to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagram instead of simple text. In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels. The pixel is the smallest addressable screen element.

Computer graphics today is largely interactive: The user controls the contents structure, and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen. Computer graphics concerns with the pictorial synthesis of real or imaginary objects from their computer based models, where the related field of image processing treats the converse process, the analysis of scenes, or the reconstruction of models of 2D or 3D objects from their pictures. The image processing can be classified as

 Image enhancement.

 Pattern detection and recognition.

 Scene analysis and computer vision.

The image enhancement deals with the improvement in the image quality by eliminating noise or by increasing image contrast. Pattern detection and recognition deals with the detection and clarification of standard patterns. And finding deviations from these patterns. The optical character recognition (OCR) technology is a practical example for pattern detection & recognition. Scene analysis deals with the recognition and reconstruction of 3D model of scene from several 2D images.

1.2 Introduction to OpenGL

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications. OpenGL is easy to learn, and it possesses most of the characteristics of other popular computer graphics system. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons.

A sophisticated library that provides these features could certainly be built on top of OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and bezier curves and surfaces. GLU is a standard part of every OpenGL implementation. Also, there is a higher-level, object-oriented toolkit, Open Inventor, which is built a top OpenGL, and is available separately for many implementations of OpenGL.

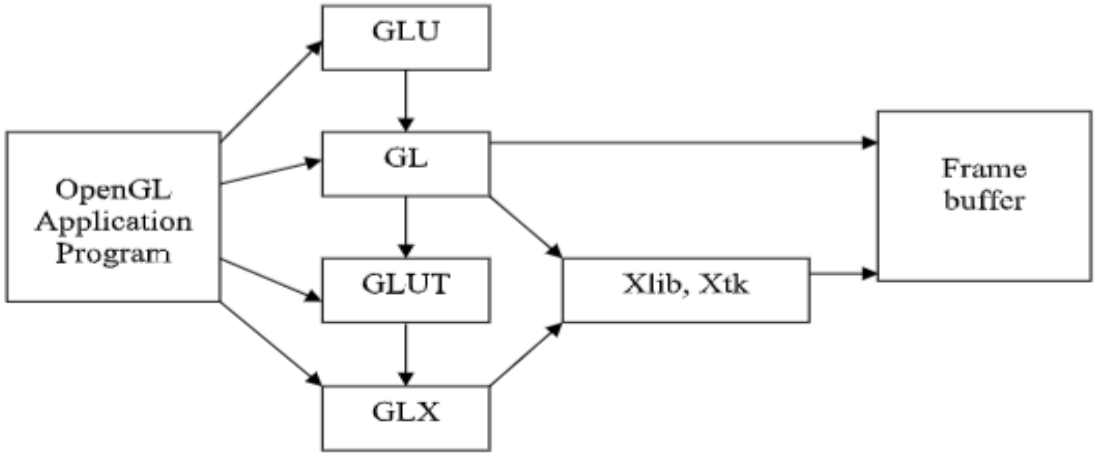

Fig. 1: Library Organization

Basically, my project is depicting a factory scene. Here we are using various OpenGL geometric primitives such as GL_LINES, GL_POLYGON. By using 2D transformations like Translation and Scaling different views of the factory scene. We have used interactions like Mouse and Menu.

1.2 Mini project Description

This project depicts a typical beverage factory like Coca-Cola where the bottles are filled and transported using OpenGL with Code Blocks IDE platform.

Initially it displays the factory which is a building with a board containing the logo of Coca-Cola and a truck which arrives to the factory. It uses mouse interaction to open the shutter of the factory This project also provides menu interactions to go to the next scene where the bottles of Coca-Cola are filled using filler. Using OpenGL primitives and API we have implemented this project.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Hardware Requirements

- ✓ Processor: Intel 486/Pentium processor or above.
- ✓ RAM : 1GB or above Storage Space.
- ✓ Monitor resolution :1000*700.

2.2 Software Requirements

- ✓ Operating System : WINDOWS 7/8/10
- ✓ Tool/IDE used: Code Blocks IDE
- ✓ Language used: C
- ✓ Compiler: GNU GCC Compiler.
- ✓ Libraries: Supporting glut32.lib, opengl32.lib & glu32.lib

.

CHAPTER 3

DESIGN

3.1 Algorithm

Step 1: Start.

Step 2: Call display function to display the scene.

Step 3: The display function calls other functions like background, building, road, truck and board.

Step 4: Add mouse interaction with left click to show the next scene where the shutter of the factory has been opened.

Step 5: Add mouse interaction with right click on menu driven by right click to see the next scene where bottles are getting filled.

Step 6: Select exit in menu by right click to terminate .

Step 7: Stop.

CHAPTER 4

IMPLEMENTATION

4.1 Output Primitive functions

glBegin (glEnum mode)

Initiates a new primitive of type mode and starts collection of vertices. Values of mode include GL_LINES, GL_POLYGON.

glEnd ()

Terminates a list of vertices.

GL_QUADS

These objects are special case of polygons. Successive groups of three and four vertices are interpreted as quadrilaterals.

In our project we used to draw conveyor belt.

glVertex ()

Specify a vertex**.**

void glVertex3f(GLfloat x, GLfloat y, GLfloat z);

In our project we used to plot every coordinate of objects.

4.2 Attribute functions

void glClearColor(GLclampf r, GLclampf g, GLclampf b, Glclampf a)

Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf floating–point numbers between 0.0 and 1.0.

void glColor3f (GLfloat red, GLfloat green, GLfloat blue)

Set the current color to object.

4.3 Geometric Transformations

void glPushMatrix () & void glPopMatrix ()

Pushes to and pops from the matrix stack corresponding to the current matrix mode.

void glTranslate [fd](TYPE x, TYPE y, TYPE z)

Translates the object from coordinates (x, y, z) to required position on Ortho In this project we used to translate the truck and conveyor belt.

void glScale [fd](TYPE sx, TYPE sy, TYPE sz)

Resize the object with coordinates sx,sy,sz.

4.4 Viewing Functions

glMatrixMode(GL_PROJECTION)

Transformation functions are incremental so we start with an identity matrix and alter it with a projection matrix that gives the view volume.

glLoadIdentity()

This ensures that each time we enter the projection modes, the matrix will be reset to the identity matrix so that the new viewing parameters are not combined with previous ones.

4.5 Input Functions

void glutCreateMenu(void (*f)(int value))

Returns an identifier for a top-level menu and registers the callback function f that returns an integer value corresponding to the menu entry selected.

In our project we added twice the menu functions to display screen of heaven and to reflect with and without snowfall effect.

void glutMouseFunc (void *f(int button, int state, int x, int y))

Registers the mouse callback function f. The callback function returns the button, the state of the button after the event (GLUT_UP, GLUT_DOWN), and the position of the mouse relative to the top left corner of the window.

4.6 Control Functions

void glutInit(int *argc, char **argv)

Initializes GLUT. The arguments from main are passed in and can be used by the application.

int glutCreateWindow(char *title)

Creates a window on the display. The string title can be used to label the window. In our project 3 windows are created as front page, way to heaven and heaven.

void glutInitDisplayMode(unsigned int mode)

Requests a display with properties in mode. The value of mode is determined by logical OR of options including the color model (GLUT_RGB,GLUT_INDEX) and buffering (GLUT_SINGLE,GLUT_DOUBLE).

void glutInitWindowSize(intwidth,int height)

Specifies the initial height and width of the window in pixel.

In our project we created window of size (1500,1000) .

void glutInitWindowPosition(intx,int y)

Specifies the initial position of the top-left corner of the window in pixel.

Our project uses Position of (0,0).

void glutMainLoop()

Cause the program to enter an event-processing loop.It should be the last statement in main.

void glutDisplayFunc(void (*func)(void))

Registers the display function func that is executed after the current callback returns.

void glutPostRedisplay()

Requests that the display callback be executed after the current

4.7 User Defined Functions

void background1()

Used to add background to the first scene. It adds ground in the background.

void building()

Used to draw building in the first scene.

void road()

Used to draw the road in the first scene on which the truck moves.

void truck(float x)

Used to draw the truck.

void board()

Used to draw the board on which the logo of the factory is written.

void cocacolalogo(float k, float l)

Used to draw logo of Coca-Cola on the bottles, board and the truck.

void background2()

Used to set background for the scene where bottles get filled.

void conveyor(float k)

Used to draw the conveyor belt on which the bottles move.

void bottle(int x, int y, float m)

Used to draw the bottles.

void filler(float x, int y)

Used to draw machine that fills the bottles with coke.

void mainmenu(int id)

Used for menu driven interaction where the scenes are selected.

void mouse(int btn,int state,int x, int y)

Used for mouse interaction to get into the second scene.

void fillcoke(int x, int y, int i, float j){

```
    float k = 0, l;
```

```
//bottle(x,y);
glColor3f(0,0,0);
//first phase
if(i<=-10){
   filling(i+2+j,55);
   for(l=0;l<=0.125*j;l=l+0.1){
      k=k+0.05;
      glBegin(GL_LINES);
         glVertex2f(x-k,y+l);
         glVertex2f(x+7.5+k,y+l);
      glEnd();
   }
}
//Second phase
else if(i<=10){
   filling(i+2+j,55);
   glBegin(GL_POLYGON);
      glVertex2f(x,y);
      glVertex2f(x+7.5,y);
      glVertex2f(x+8.75,y+2.5);
      glVertex2f(x-1.25,y+2.5);
   glEnd();
   for(l=0;l<=0.4*j;l=l+0.1){
```

```
            k=k+0.009;
            glBegin(GL_LINES);
                glVertex2f(x-1.25+k, y+2.5+l);
                glVertex2f(x+8.75-k, y+2.5+l);
            glEnd();
        }
    bottle(x, y, 1);
}
void display()
{
    int i,l;
    float j,k;
    if(flag > 2)
        flag = 1;
    if(flag < 1)
        flag = 2;
    if(flag == 1){
        glClearColor(0.6,0.85,0.9176,1);
        background1();
        building();
        road();
        truck(k);
        board();
```

```
        k=k+0.04;
        if(k>100) k=-35;
 else if(flag == 2){
        mouseon=0;
        j=j+speed;
        glClearColor(0.7,0,0,1);
        background2();
      conveyor(j);
        for(i=-10;i<=90;i=i+20)
            fillcoke(i+j,10,i,j);
        for(i=-13;i<100;i=i+20)
            filler(i+j,75);
        if(j>20) j=0;
        glutSwapBuffers();
    }
    glFlush();
}
```

CHAPTER 5

RESULTS

5.1 SCREENSHOTS

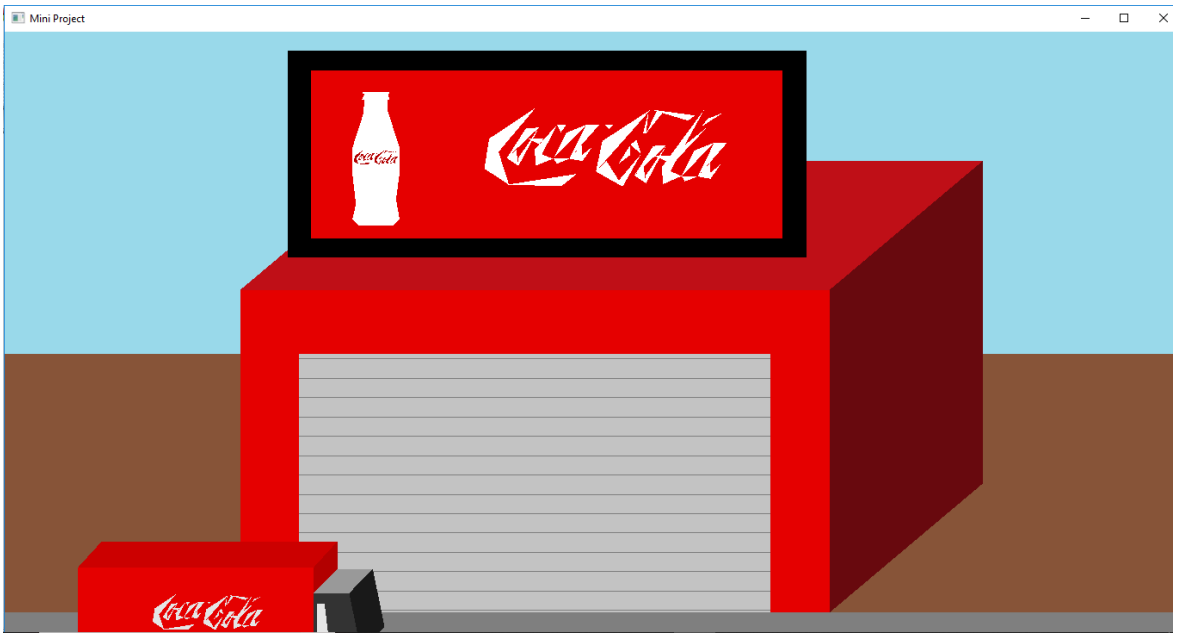

Figure 5.1 Outside Factory

This is the first scene where we can see the Coca-Cola factory and a truck with the Coca-Cola logo which is passing by the factory on the road.

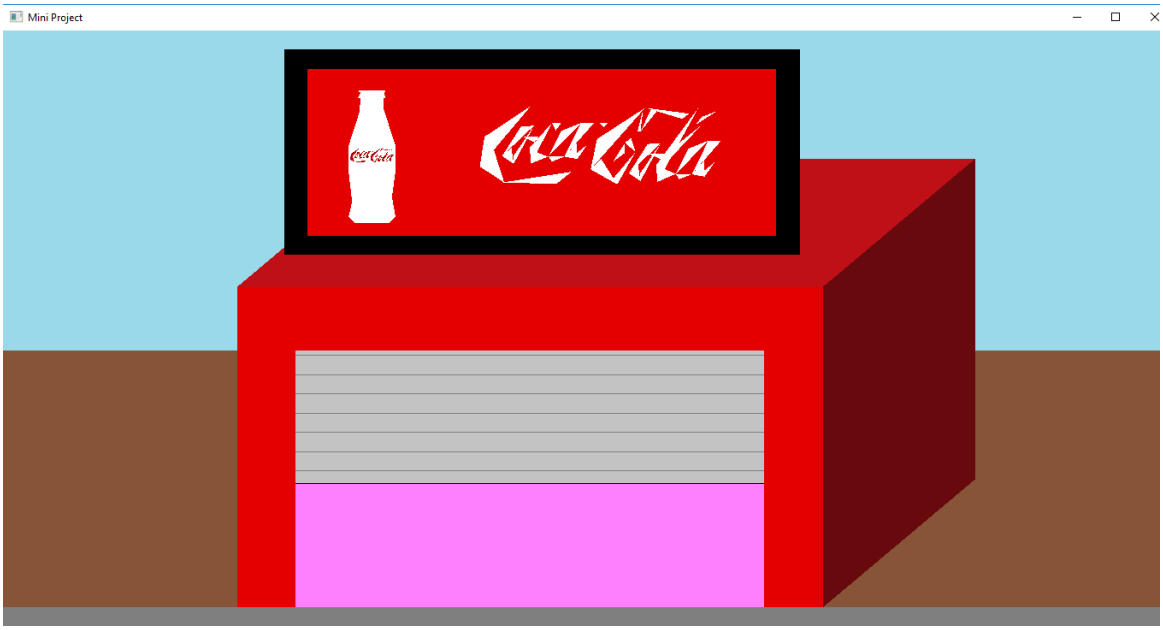

Figure 5.2: Opening the shutter

This is the second scene which appears on the left click of the mouse where the shutter of the factory is open.

Figure 5.3: Filling of bottles

This is the third scene in which the bottles with Coca-Cola logo gets willed with coke with the help of filler. It basically has five phases using which the bottles are filled

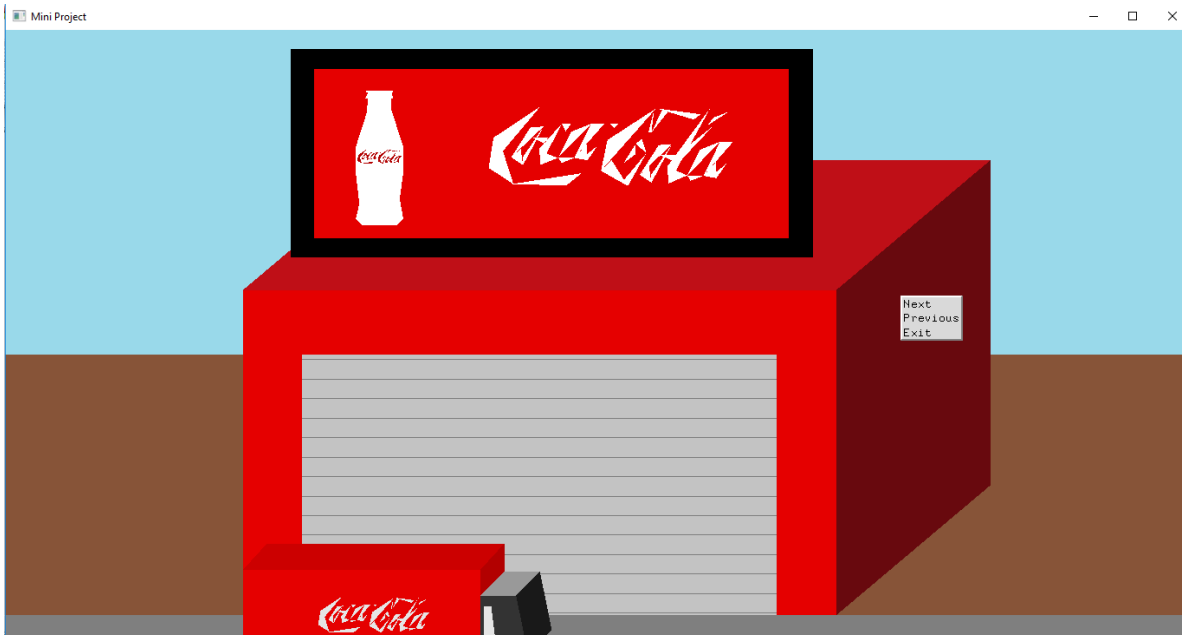

Figure 5.4: closed factory

This is scene in which the shutter is closed and using exit in menu we can exit the project.

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENTS

The concept of Coca-Cola factory is to visualize a typical beverage factory. From this project we have learnt the implementation of OpenGL primitives, menu interactions, mouse interactions. Initially this started with idea of a factory where manufacturing of beverage like Coca-Cola is done and different stages of the transportation and filling the bottles of Coca-Cola is done and the scenes of this is visualized using this project. As the implementation carried away it came out with the five phases of filling the bottles and interactions where added.

The scenario can be extended with further additions as follows:

- Incoming of empty bottles on conveyor belt can be shown before filling the bottles.
- The filled bottles with the caps on it moving on the conveyor belt can be shown once the bottles are filled.
- After filling of the bottles the packaging of the bottles in the crates can be shown before transporting it.

REFERENCES

The books and the other sources we referred while doing this project are as follows:

Referred Books

- Donald Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 4th Edition, Pearson Education, 2011.
- Edward Angel: Interactive Computer Graphics-A Top Down approach with OpenGL, 5th Edition, Pearson Education, 2008.

Websites Referred

- https://www.stackoverflow.com
- http://www.opengl-tutorial.org/beginners-tutorials/
- https://www.opengl.org/
- https://learnopengl.com/Introduction