CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is a sub-field of computer science and is concerned with digitally synthesizing and manipulating visual content. Although the term refers to three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing. Computer graphics is often differentiated from field of visualization, although two have some similarities. Graphics are visual presentation on some surface like wall, canvas, computer screen. Graphics often combine text, illustration and colour.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural and even conceptual structures. Computer graphics today is largely interactive. The user controls the contents, structure and appearance of objects and of their displayed images by using input devices such as keyboard, mouse or touch-sensitive panel on screen.Figure1.1 shows the library organization of openGL.

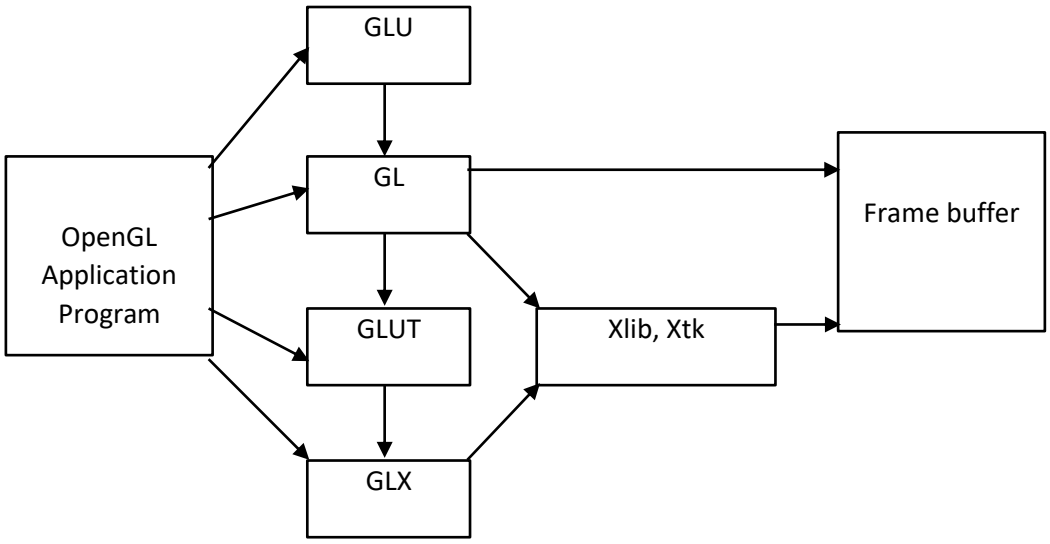

Figure 1.1: OpenGL Library Organization

Computer graphics is no more a rarity. Even people who do not use computers in their daily work encounter computer graphics in television commercials and as cinematic special effects. Computer graphics is a part of all user interfaces and is indispensable for visualizing objects.

Graphical interfaces have replaced textual interfaces as the standard means for user-computer interaction. Graphics has also become a key technology for communication ideas, data and trends in most areas of commerce, science, engineering and education. Much of the task of creating effective graphic communication lies in modelling the objects whose image we want to produce.

1.2 OpenGL

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

Silicon Graphics Inc., (SGI) started developing OpenGL in 1991 and released it in January 1992; applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006 OpenGL has been managed by the non-profit technology consortium Khronos Group.

One aspect of OpenGL that suits it so well for use in computer graphics course is its device independence or portability. A student can develop and run program on any available computer. OpenGL offers rich and highly usable API for 2D graphics and image manipulation, but its real power emerges with 3D graphics.

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to simplify the programming tasks, including the following:

The OpenGL Utility Library(GLU)contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. The library is provided as part of every OpenGL implementation.

Function names in the OpenGL basic library are prefixed with gl, and each component word within a function name has its first letter capitalized **glBegin,glClear,**

glCopyPixels,glPolygonMode. Component words within a constant name are written in capital letters, and

the underscore (_) is used as a separator between all component words in the name. GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE OpenGL data-types GLbyte, GLshort, GLint, GLfloat, GLdouble, Glboolean.

The **OpenGL Utility Library (GLU)** contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. The library is provided as

part of every OpenGL implementation [2].

OpenGL Utility Toolkit (GLUT)

Provides a library of functions for interacting with any screen-windowing system .The GLUT library functions are prefixed with glut contains methods for describing and rendering quadric curves and surfaces. GLUT is an interface to other device-specific window systems, so the programs will be device-independent [2].

1.3 About Project

Our project basically includes all the simple and basic functions that we have studied and have implemented it in the project. The mouse function and the menu options together help us to develop a digital canvas.

The mouse functions help in drawing in our project. We have used all the basic functions like the Translate, Rotate, and Scaling functions along with some useful algorithms like the circle algorithm to draw circles, lines, rectangles and more.

In the project the canvas is considered as matrix where operations can be performed to create designs, we take in the mouse point where it clicks and depending on that and mode of choice i.e. line, circle, brush, we draw the necessary pixels on the canvas.

CHAPTER 2

REQUIREMENT SPECIFICATIONS

The basic purpose of software requirement specification (SRS) is to bridge the communication between the parties involved in the development project. SRS is the medium through which the user's needs are accurately specified; indeed, SRS forms the basis of software development. Another important purpose of developing an SRS is helping the users understand their own needs.

Now we will be discussing the requirement analysis of the project. This report gives the description of the roles of users, the functional overviews of the project, input and output characteristics and also the hardware and software for the project.

2.1 Hardware Requirements

- Processor-Intel or AMD (Advanced Micro Devices)
- RAM-512MB(minimum)
- Hard Disk -1MB(minimum)
- Mouse
- Keyboard
- Monitor

2.2 Software Requirements

- C++ language compilers.
- OpenGL libraries.
- WINDOWS or LINUX based platform OS.
- gedit.

CHAPTER 3

SYSTEM DESIGN

3.1 FLOWCHART

A flowchart is a common type of chart that represents an algorithms or process showing the steps as boxes of various kinds and their order by connecting these with arrows. The figure 3.1 shows the flow of control in our project

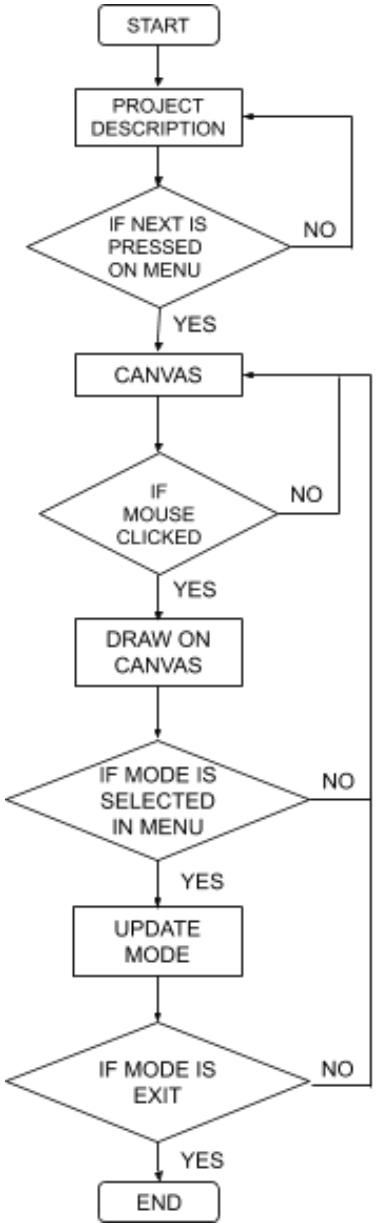

Figure 3.1: Flowchart for the project

The given flowchart represents the flow in which our program executes. The project begins with a display of the project title and the names of the students and faculty who have helped build this. Here we give a mouse-click triggered menu from which a user can select to continue to the main canvas. Following this we come to an open white canvas in which we have defaulted a red brush which draws when you drag or click with your cursor.

Here, we allow multiple modes in which each mode lets us draw something different i.e. lines, circles, rectangles, brush etc. In each mode we have given specific functions to follow. For instance, when line mode is selected the program waits for the user to select two points after which it applies Bresenham's line drawing algorithm to draw the line.

The menu also specifies a colour choice which can also be set to random, in which before drawing it chooses a random colour from the choices. One interesting point is that in these there is an eraser option. As the background is white, the eraser is nothing but a white brush that overwrites any other pixels painted on the screen.

Finally, we have also given the choices of clearing the entire screen or quitting the program.

CHAPTER 4

IMPLEMENTATION

The implementation stage of the model involves the following phases.

- Implementation of OpenGL built in function.
- User defined function Implementation.

4.1 OpenGL Functions

4.1.1 Specifying Simple Geometry

void glBegin (glEnum mode)

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

void glEnd()

Terminates a list of vertices.

4.1.2 Attributes

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 glPointSize(GLfloat size)

Sets the point size attribute in pixels.

4.1.3 Working with the window

void glFlush()

Forces any buffered OpenGL commands to execute.

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.

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(int width,int height)

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

void glutInitWindowPosition(int x,int y)

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

void glutMainLoop()

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

void glutLeaveMainLoop()

Causes the program to exit the event-processing loop.

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

Registers the display function func that is executed after the current call-back returns.

void glutPostRedisplay()

Requests that the display call-back be executed after the current call-back returns.

4.1.4 Interactions

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.

void glutKeyboardFunc(void *f(char key, int width, int height)

Registers the keyboard call-back function f. The call-back function returns the ASCII code of the key pressed and the position of the mouse.

4.1.5 Enabling Features

void glEnable(GLenum feature)

Enables an openGL feature. Features that can be enabled include GL_DEPTH_TEST, GL_LIGHTING, GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D, GL_LINE_SMOOTH, GL_POLYGON_SMOOTH, GL_POINT_SMOOTH, GL_BLEND, GL_LINE_STIPPLE, GL_POLYGON_STIPPLE, GL_NORMALIZE.

void glDisable(GLenum feature)

Disables an openGL feature.

4.1.6 Transformations

void glMatrixMode(GLenum mode)

Specifies which matrix will be affected by subsequent transformations. Mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.

void glLoadIdentity()

Sets the current transformation matrix to an identity matrix.

void glPushMatrix() & void glPopMatrix()

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

void glRotate[fd](TYPE angle, TYPE dx, TYPE dy, TYPE dz)

Alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz).

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

Alters the current matrix by a displacement of(x, y, z).

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

Alters the current matrix by a scaling of(sx, sy, sz).

4.1.7 Viewing

void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)

Defines an orthographic viewing volume with all parameters measured from the centre of projection plane.

4.2 Working with the window system

void glFlush();

Forces any buffered OpenGL command to execute.

void glutInit(intargc,char **argv);

glutInit() should be called before any other glut routine, because it initializes the GLUT library. glutInit() will also process command line options, but he specific options are window system dependent.

int glutCreateWindow(char *title);

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

4.3 User defined functions

We have used many user defined functions for the convenience in translating and using other basic functions.

void display(void)

clears the canvas background to white and displays the points stored in Dot class.

void clear(void)

clears the canvas and deletes the Dot class objects.

void quit(void)

used to exit the canvas.

void drawDot(int mousex, int mousey)

creates new Dot object based on mouse (x, y) coordinates.

void drawBrush(int x, int y)

randomly draws dots around the cursor position.

void drawLine(int x1, int y1, int x2, int y2)

used to draw line using Bresenham's algorithm.

void drawRectangle(int x1, int y1, int x2, int y2)

used to draw a circle.

void drawCircle(int x1, int y1, int x2, int y2)

used to create circle using midpoint algorithm.

void erase(int x, int y)

used to paint white at the location of mouse click.

void keyboard(unsigned char key, int xIn, int yIn)

adds keyboard functionalities such as to quit, clear, increase or decrease brush size.

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

used to get the (x, y) coordinates of mouse click and to draw shapes on canvas.

CHAPTER 5

RESULTS

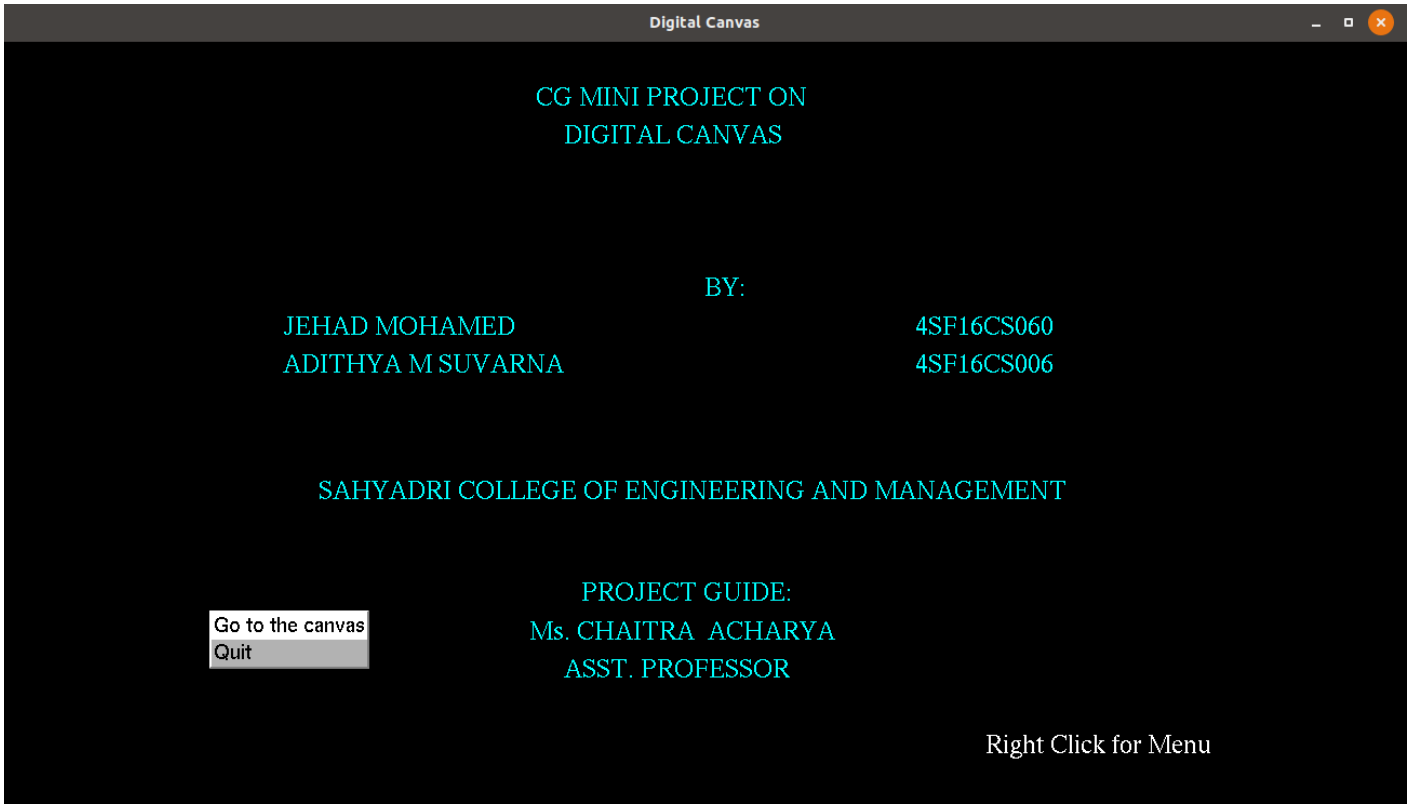

Figure 5.1: Introduction page

Figure 5.1 depicts the initial scene of the project which includes all the details about the project members, the guides and the title of the project.

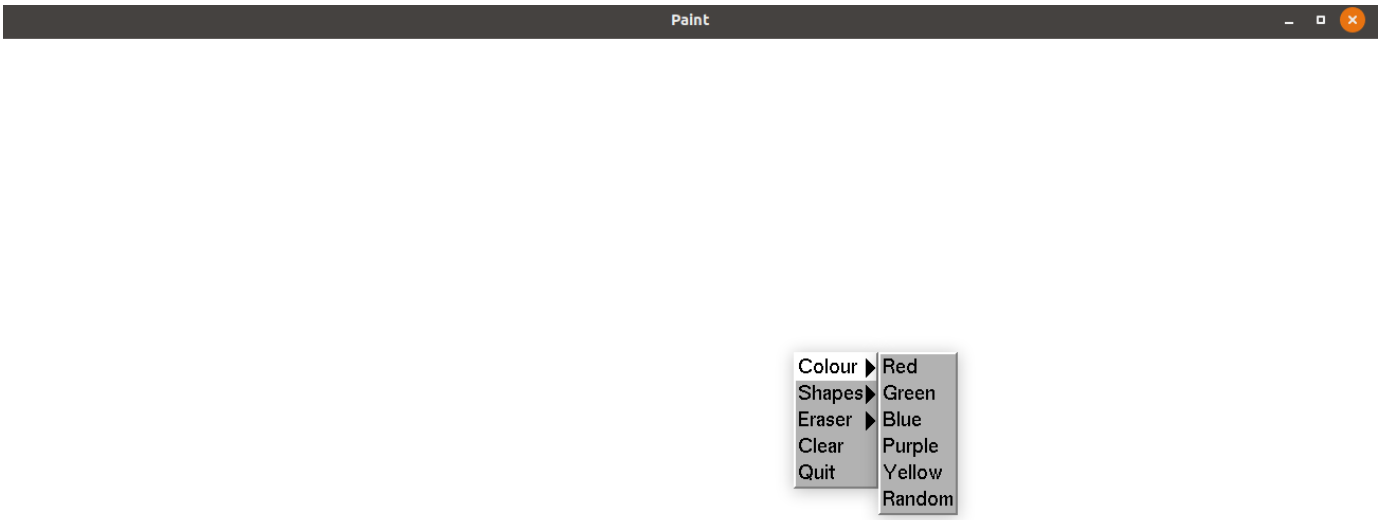

Figure 5.2: Colour selection

Figure 5.2 displays the canvas with a menu option Colour selected. Here you can choose different colours for drawing.

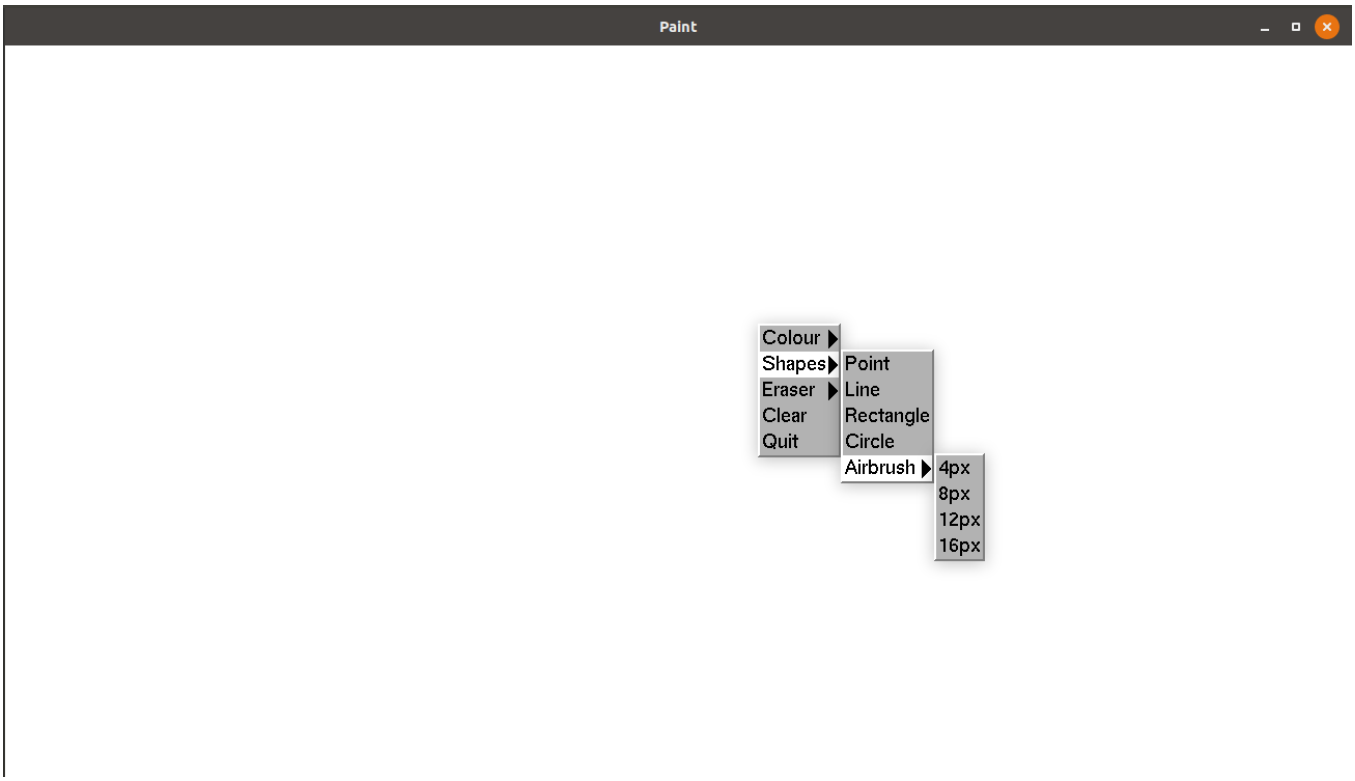

Figure 5.3: Shape selection

Figure 5.3 shows sub menu where you can choose different shapes, i.e Point, Rectangle, Circle, Airbrush.

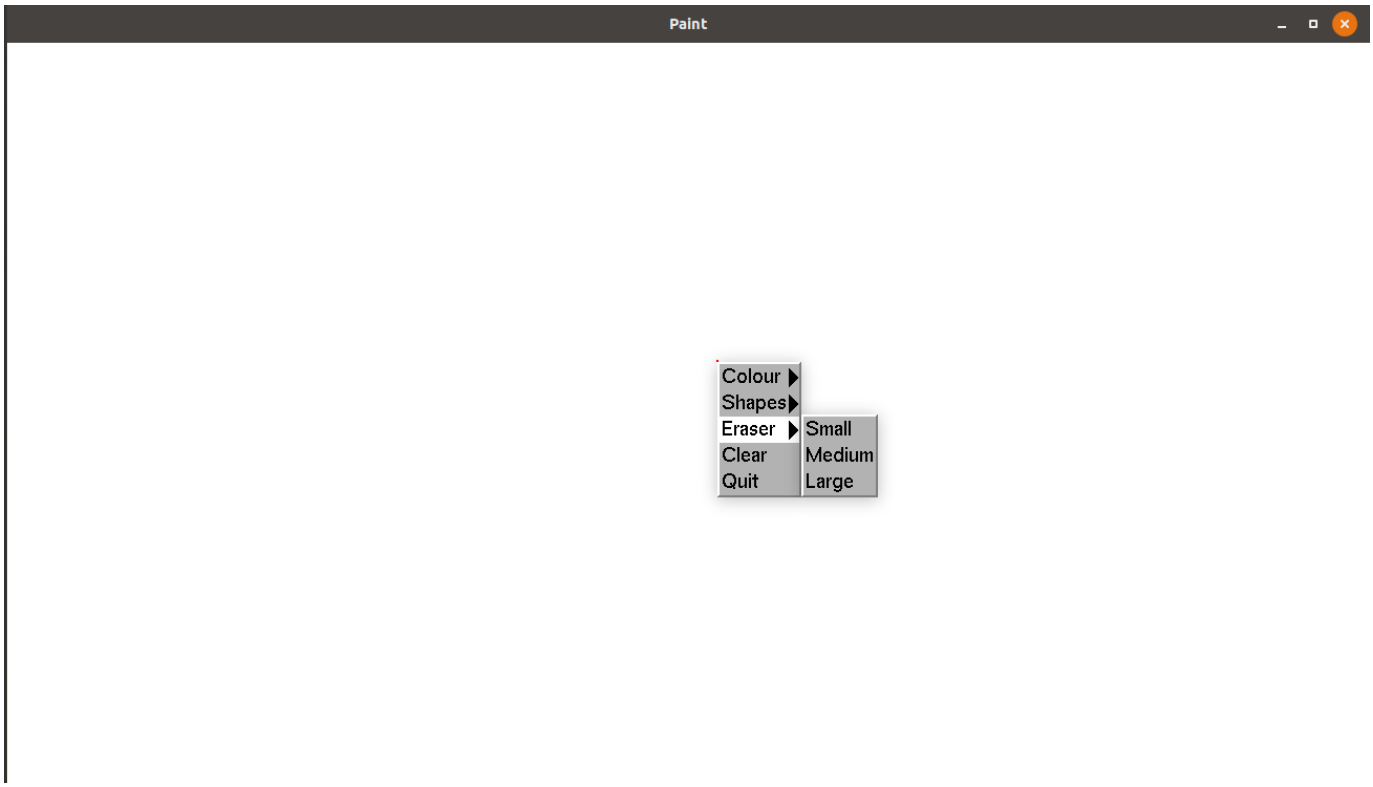

Figure 5.4: Eraser Mode

Figure 5.4 shows eraser menu where you can choose different size of the eraser.

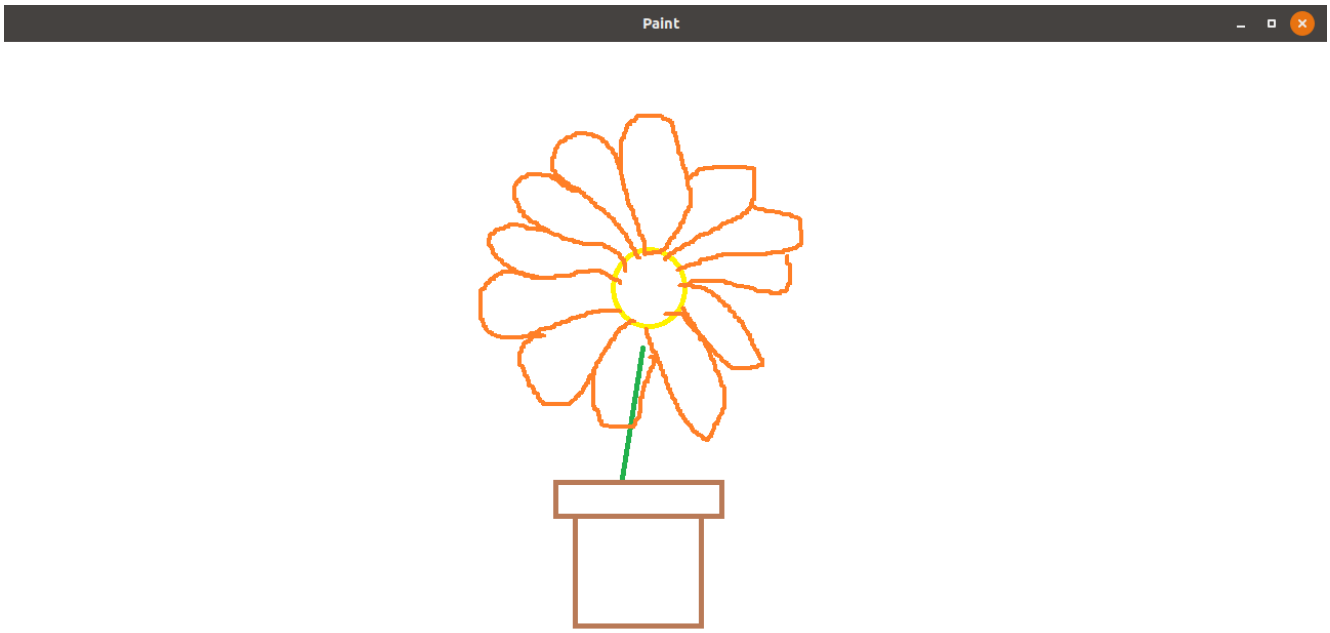

Figure 5.5: Canvas with a flower using shapes drawn

Figure 5.5 shows the canvas with a flower made of different shapes like rectangle, points, circles, lines drawn on it.

CHAPTER 6

CONCLUSION

Our project **"Digital Canvas"** has helped us in understanding about computer graphics and OpenGL basic functions which can be used to manipulate the data and provide some animations and various concepts and methodologies used in computer graphics.

The project illustrates the working of digital canvas. It helped us learn more about the subject practically and implement it in this project to demonstrate some simple movements of the objects, hidden surface removal problems, and many functions which is useful in depicting some of the basic concepts in our day today life.

Developing this project helped us to learn more about the computer graphics practically and to add interactive functions required for the project. The project is user friendly and has the features which can be easily understood by any user. It demonstrates the OpenGL applications in 3D with animation. The program has been written in C++ language with OpenGL libraries.

The project is used as an informative tool or a program depicting the basic working of any paint program or digital canvas. The project is used to demonstrate the uses of computer graphics and OpenGL functions in a very informative and colourful way. The project can be extended further by adding all the functions required for 3D such as lighting, shadows and materials.

REFERENCES

[1] http://freeglut.sourceforge.net/docs/api.php

[2] www.opengl.org

[3] Interactive Computer Graphics A Top-Down Approach with OpenGL -Edward Angel, 5th Edition, Addison-Wesley, 2008.

[4] James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, **Computer Graphics**, Pearson Education 1997.

[5] OpenGL Super Bible: Comprehensive Tutorial and Reference (5th Edition).

[6] F.S. Hill Jr**.: Computer Graphics using OpenGL**, 3rd Edition, PHI, 2009.