**CHAPTER 4**

**DESIGN AND IMPEMENTATION**

**4.1 Header Files Used**

The header files used are:

1. **#include<stdio.h>**: This header in the C standard library contains macro definition, constants and declaration of function and types used for various standard input output operations. The name stdio stands for standard input output.
2. **#include<stdlib.h>**: It is the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversion and others. its compatible with C++ and known as cstd.lib in C++.The name stdlib stands for standard library.
3. **#include<GL/glut.h>**: It is the library of utilities for OpenGL programs which primarily performs system level input output with the host operating system. Functions performed include window definition, window control and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode)are also provided. Glut even has some limited support for creating pop-up menus.

## 4.2 OpenGL API’s Used

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

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

* **void glutInitDisplayMode(unsigned int mode)**

Requests a display with the properties in the mode; the value of mode is determined by the logical OR of options including the color model (GLUT\_RGB, GLUT\_INDEX) and buffering (GLUT\_SINGLE, GLUT\_DOUBLE).

* **void glutCreateWindow(char \*title)**

Creates a window on 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.

* **void glutInitWindowSize(int width, int height)**

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

* **void glutInitWindowPosition(int x, int y)**

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

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

Registers the display function func that is executed when the window needs to be redrawn.

* **void CreateMenu(void)**

This function is used to create menus which are used as options in program.

* **void glutReshapeFunc(void \*f(int width, int height))**

Registers the reshape callback function f. the callback function returns the height and width of the new window. The reshape callback invokes the display callback.

* **void glutMainLoop()**

Causes the program to enter an event-processing loop.

* **void glutSwapBuffers()**

Swaps the front and back buffers.

* **void glColor3[b I f d ub us ui](TYPE r, TYPE g, TYPE b)**

Sets the present RGB colors. Valid types are bytes(b), int(i), float(f), double(d), unsigned byte(ub), unsigned short(us), and unsigned int(ui). The maximum and minimum value for floating point types are 1.0 and 0.0 respectively, whereas the maximum and minimum values of the discrete types are those of the type, for eg, 255 and 0 for unsigned bytes.

* **void glClear(glEnum mode)**

Clears the buffers namely color buffer and depth buffer mode refers to GL\_COLOR\_BUFFER\_BIT or DEPTH\_BUFFER\_BIT.

* **void glMatrixMode(GLenum mode)**

Specifies which matrix will be affected by subsequent transformations, Mode can be GL\_MODELVIEW or GL\_PROJECTION.

* **void glLoadIdentity( )**

Sets the current transformation matrix to identity matrix.

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

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

* **void glClearColor(GLclampf r, GLclampf g, GLclamp b, Glclamp a)**

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

* **void glViewport(int x ,int y, GLsizei width, GLsizei height)**

Specifies the width\*height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.

* **void gluLookAt(GLdouble eyex,GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble aty, GLdouble atz, GLdouble upx, GLdouble upy, GLdouble upz)**

Postmultiplies the current matrix determined by the viewer at the eye point looking at the point with specified up direction.

**4.3 User Defined Functions**

* **int main( ) :**

The main function is used for creating the window for display of the model of the atom. Here, we create menu for ease of use for the user. The callback functions, i.e., myIdle callback, myinit callback, display callback, are written in main.

* **myinit():**

This function is used for initialization of square. In selection mode, each square is given two names one for the row and the other for the column on the grid. The color of each square is determined by its position on the grid.

void init(void)

{

srand((unsigned)time(NULL));

int i, j;

for (i = 0; i < 6; i++)

for (j = 0; j < 6; j ++)

board[i][j] = (rand())%3;

glClearColor (1.0, 1.0, 1.0, 1.0);

}

**myDisplay( ) :**

This function creates and translates all the objects in a specified location in a particular order.

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POINTS); /\* render all particles \*/

for(i=0;i<num\_particles;i++)

{

glColor3fv(color[particles[i].color]);

glVertex3fv(particles[i].position);

}

glEnd();

glColor3f(0.0,0.0,0.0);

glutWireCube(2.2); /\* outline of box \*/

glutSwapBuffers();

}

* **void Reshape ( ) :**

It is used for defining the viewport volume using glViewport(int x,int y, GLsizei width,Glsizei height).

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0.0, 6.0, 0.0, 6.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

* **void keyboard( ) :**

This function is used to define the keyboard function of the given project.

void keyboard(unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(0);

break;

}

}

* **void pickSquare( ) :**

This function is used for checking the square is of same color or not which helps to play the game and we also defined the pop-up menu options of the program.

void pickSquares(int button, int state, int x, int y)

{

GLuint selectBuf[BUFSIZE];

GLint hits;

GLint viewport[4];

if(end!=1)

{

if(button==GLUT\_RIGHT\_BUTTON || state==GLUT\_DOWN)

{

glutCreateMenu(color\_menu);

glutAddMenuEntry("show score",1);

glutAddMenuEntry("undo",2);

glutAddMenuEntry("continue",3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutPostRedisplay();

}

if (button != GLUT\_LEFT\_BUTTON || state != GLUT\_DOWN)

{

return;

}

glGetIntegerv (GL\_VIEWPORT, viewport);

glSelectBuffer (BUFSIZE, selectBuf);

(void) glRenderMode (GL\_SELECT);

glInitNames();

glPushName(0);

glMatrixMode (GL\_PROJECTION);

glPushMatrix ();

glLoadIdentity ();

/\* create 5x5 pixel picking region near cursor location \*/

gluPickMatrix ((GLdouble) x, (GLdouble) (viewport[3] - y),

5.0, 5.0, viewport);

gluOrtho2D (0.0, 6.0, 0.0, 6.0);

drawSquares (GL\_SELECT);

glMatrixMode (GL\_PROJECTION);

glPopMatrix ();

glFlush ();

hits = glRenderMode (GL\_RENDER);

processHits (hits, selectBuf);

glutPostRedisplay();

}

}