**Batch: D - 1 Roll No.: 16010122096**

**Experiment No. 10**

|  |
| --- |
| **TITLE**: WAP to implement Simple Interaction with the mouse and keyboard. |

**AIM:**

Write an OpenGL program to demonstrate use of interaction through mouse and keyboard.

(Example: Pressing ‘p’ draws a dot at the mouse position; pressing the left arrow key adds a point to some (global) list, but does no drawing; pressing ‘E’ exits from the program.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO4: Understand the computer Input & interaction, Curves and Computer Animation

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm/ Pseudocode for each process:**

**Initialize variables:**

* **Create a list points to store coordinates of points.**
* **Set window width winWidth and height winHeight.**

**Define display function:**

* **Clear the screen.**
* **Begin drawing points.**
* **Loop through points and draw each point.**
* **End drawing and swap buffers to display the updated content.**

**Define mouse callback function:**

* **If the left mouse button is clicked:**
  + **Convert the mouse coordinates to OpenGL coordinates.**
  + **Add the converted coordinates to points.**
  + **Request a screen redraw.**

**Define keyboard callback function:**

* **If the key 'p' is pressed:**
  + **Convert the mouse coordinates to OpenGL coordinates.**
  + **Add the converted coordinates to points.**
  + **Request a screen redraw.**
* **If the key 'E' or 'e' is pressed:**
  + **Exit the program.**

**Define special keyboard callback function:**

* **If the left arrow key is pressed:**
  + **Add a point at the origin (0, 0) to points.**

**Define initialization function:**

* **Set the background color to white.**
* **Set the point color to black.**
* **Set the point size to 5.**
* **Set up an orthographic projection (-1 to 1 in both x and y axes).**

**Main function:**

* **Initialize the OpenGL environment.**
* **Create a window with size winWidth x winHeight.**
* **Call the initialization function.**
* **Register the display, mouse, keyboard, and special keyboard callback functions.**
* **Enter the event-processing loop.**

**Implementation details:**

*#include* <GL/glut.h>

*#include* <vector>

std::vector<float> points;

int winWidth = 800, winHeight = 600;

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glBegin(GL\_POINTS);

*for* (size\_t i = 0; i < points.size(); i += 2) {

        glVertex2f(points[i], points[i + 1]);

    }

    glEnd();

    glutSwapBuffers();

}

void mouse(int *button*, int *state*, int *x*, int *y*) {

*if* (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

        float xPos = (float)x / winWidth \* 2 - 1;

        float yPos = 1 - (float)y / winHeight \* 2;

        points.push\_back(xPos);

        points.push\_back(yPos);

        glutPostRedisplay();

    }

}

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

*if* (key == 'p') {

        float xPos = (float)x / winWidth \* 2 - 1;

        float yPos = 1 - (float)y / winHeight \* 2;

        points.push\_back(xPos);

        points.push\_back(yPos);

        glutPostRedisplay();

    } *else* *if* (key == 'E' || key == 'e') {

        exit(0);

    }

}

void specialKeyboard(int *key*, int *x*, int *y*) {

*if* (key == GLUT\_KEY\_LEFT) {

        points.push\_back(0.0f);

        points.push\_back(0.0f);

    }

}

void init() {

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    glColor3f(0.0f, 0.0f, 0.0f);

    glPointSize(5.0f);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

}

int main(int *argc*, char\*\* *argv*) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

    glutInitWindowSize(winWidth, winHeight);

    glutCreateWindow("OpenGL Interaction Example");

    init();

    glutDisplayFunc(display);

    glutMouseFunc(mouse);

    glutKeyboardFunc(keyboard);

    glutSpecialFunc(specialKeyboard);

    glutMainLoop();

*return* 0;

}

**Output(s) (Screen Shot):**

****

**Conclusion and discussion:**

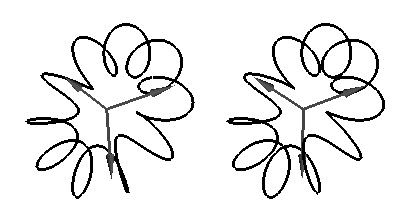
The program successfully demonstrates interaction using mouse and keyboard inputs in OpenGL, achieving dynamic point creation and exit functionality**.**

**Date:**

**Signature of faculty in-charge**

**Post lab Question**

**Write a program to draw the following**



#include <GL/glut.h>

#include <cmath>

const int POINTS = 1000;  // Number of points on the flower curve

const float PI = 3.14159265358979323846f;

// Function to draw the parametric flower curve

void drawFlowerCurve() {

    glColor3f(0.0f, 0.0f, 0.0f);

    glLineWidth(2.0f);

    // Start drawing the curve

    glBegin(GL\_LINE\_LOOP);

    for (int i = 0; i < POINTS; ++i) {

        float t = (2 \* PI \* i) / POINTS; // Parametric angle

        float x = (0.3 \* cos(3 \* t)) \* cos(t);  // Parametric equation for x

        float y = (0.3 \* cos(3 \* t)) \* sin(t);  // Parametric equation for y

        float z = 0.1 \* sin(2 \* t);             // Parametric equation for z

        glVertex3f(x, y, z);

    }

    glEnd();

}

// Display function for rendering

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glLoadIdentity();

    gluLookAt(0.0, 0.0, 2.0,  // Camera position

              0.0, 0.0, 0.0,  // Look at the origin

              0.0, 1.0, 0.0); // Up vector

    // Draw flower curve

    drawFlowerCurve();

    glutSwapBuffers();

}

// Initialization function

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0); // White background

    glEnable(GL\_DEPTH\_TEST);

}

// Main function

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(800, 600);

    glutCreateWindow("Flower-like Curve");

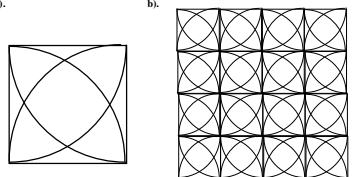
    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

I

*#include* <GL/glut.h>

*#include* <cmath>

const int NUM\_SEGMENTS = 100;

const float PI = 3.14159265358979323846;

const int GRID\_SIZE = 4;

void drawArc(float *cx*, float *cy*, float *radius*, float *startAngle*, float *endAngle*) {

    glBegin(GL\_LINE\_STRIP);

*for* (int i = 0; i <= NUM\_SEGMENTS; i++) {

        float theta = startAngle + (endAngle - startAngle) \* (float)i / (float)NUM\_SEGMENTS;

        float x = radius \* cos(theta);

        float y = radius \* sin(theta);

        glVertex2f(cx + x, cy + y);

    }

    glEnd();

}

void drawSquareWithArcs(float *xOffset*, float *yOffset*, float *size*) {

    float radius = size;

*// Draw square*

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(xOffset, yOffset);

    glVertex2f(xOffset + size, yOffset);

    glVertex2f(xOffset + size, yOffset + size);

    glVertex2f(xOffset, yOffset + size);

    glEnd();

*// Bottom left arc*

    drawArc(xOffset, yOffset, radius, 0.0f, PI / 2.0f);

*// Bottom right arc*

    drawArc(xOffset + size, yOffset, radius, PI / 2.0f, PI);

*// Top right arc*

    drawArc(xOffset + size, yOffset + size, radius, PI, 3.0f \* PI / 2.0f);

*// Top left arc*

    drawArc(xOffset, yOffset + size, radius, 3.0f \* PI / 2.0f, 2.0f \* PI);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0, 0, 0);

    float size = 0.5f;

*// Draw the grid of squares with arcs*

*for* (int i = 0; i < GRID\_SIZE; i++) {

*for* (int j = 0; j < GRID\_SIZE; j++) {

            float xOffset = -1.0f + i \* size;

            float yOffset = -1.0f + j \* size;

            drawSquareWithArcs(xOffset, yOffset, size);

        }

    }

    glFlush();

}

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

int main(int *argc*, char\*\* *argv*) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutCreateWindow("Grid of Squares with Arcs");

    init();

    glutDisplayFunc(display);

    glutMainLoop();

*return* 0;

}