Batch: D-2 **Roll No.:** 16010122151

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:

CO1: Understand the basic concepts of computer graphics and OpenGL

CO4: Understand the basics and working of Visualization

Books/ Journals/ Websites referred:

OpenGL Archives - GeeksforGeeks
Getting started with OpenGL - GeeksforGeeks

Algorithm/ Pseudocode for each process:

- **glutDisplayFunc(myDisplay);** Whenever the system determines that a screen window should be redrawn it issues a "redraw" event. This happens when the window is first opened, and when the window is exposed by moving another window off of it.
- **glutReshapeFunc(myReshape);** Screen windows can be reshaped by the user, usually by dragging a corner of the window to a new position with the mouse. (Simply moving the window does not produce a reshape event.) As we shall see, myReshape() is automatically passed arguments that report the new width and height of the reshaped window.
- **glutMouseFunc(myMouse);** When one of the mouse buttons is pressed or eleased a mouse event is issued. myMouse() is automatically passed arguments that describe the mouse location and the nature of the button action.
- **glutKeyboardFunc(myKeyboard);** This registers the function myKeyboard() with the event of pressing or releasing some key on the keyboard. myKeyboard() is automatically passed arguments that tell which key was pressed. Conveniently, it is also passed data as to the location of the mouse at the time the key was pressed.

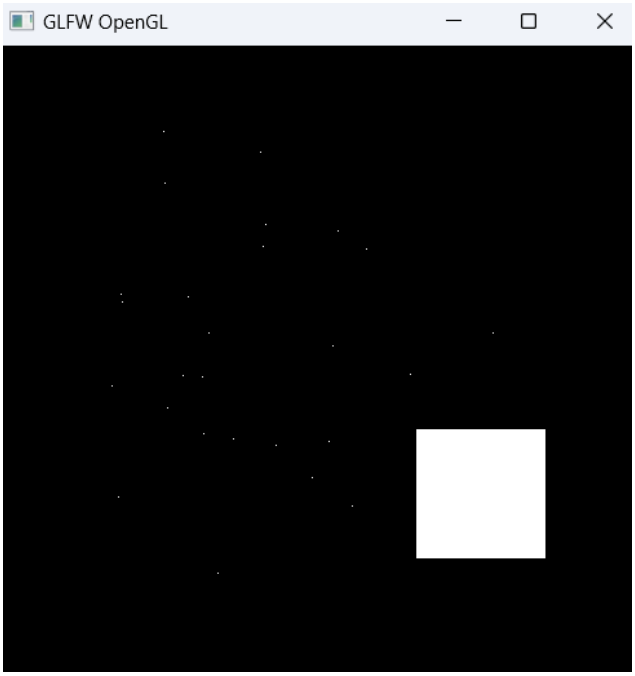
Implementation details:

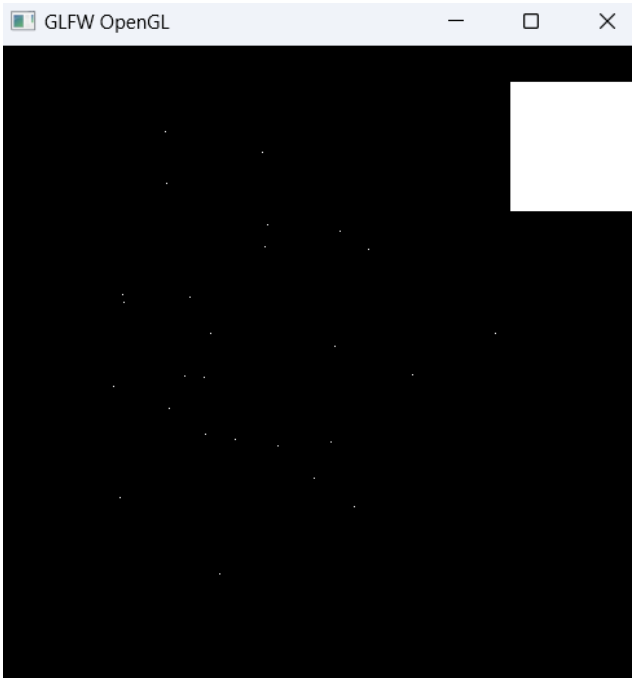
```
import glfw
from OpenGL.GL import *
from OpenGL.GLU import gluOrtho2D

# Global variables to store points and their positions
points = []  # List to store points
squarePosX = 0.0
squarePosY = 0.0
squareSize = 0.2

# Function to render the square and points
def draw_square():
    glClear(GL_COLOR_BUFFER_BIT)

    # Draw the square
    glPushMatrix()
    glTranslatef(squarePosX, squarePosY, 0.0)
    glBegin(GL_QUADS)
    glVertex2f(-squareSize, -squareSize)
    glVertex2f(squareSize, -squareSize)
    glVertex2f(squareSize, squareSize)
    glVertex2f(-squareSize, squareSize)
    glEnd()
    glPopMatrix()

    # Draw all points stored in the points list
```

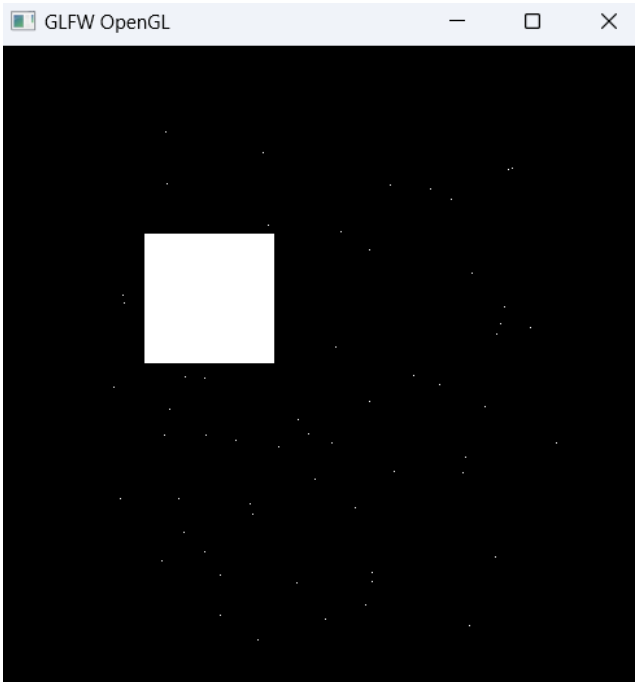
```
    glBegin(GL_POINTS)
    for point in points:
        glVertex2f(point[0], point[1])
    glEnd()

    # Flush and swap buffers
    glFlush()

# Keyboard callback
def key_callback(window, key, scancode, action, mods):
    global squarePosX, squarePosY, points
    if action == glfw.PRESS:
        if key == glfw.KEY_W: squarePosY += 0.05
        if key == glfw.KEY_S: squarePosY -= 0.05
        if key == glfw.KEY_A: squarePosX -= 0.05
        if key == glfw.KEY_D: squarePosX += 0.05

        if key == glfw.KEY_P:  # Draw a dot at the mouse position
            xpos, ypos = glfw.get_cursor_pos(window)
            window_width, window_height = glfw.get_window_size(window)
            point_x = (xpos / window_width) * 2 - 1
            point_y = 1 - (ypos / window_height) * 2
            points.append((point_x, point_y))  # Add point to the list

        if key == glfw.KEY_LEFT:  # Add a point to the global list (no
drawing)
            xpos, ypos = glfw.get_cursor_pos(window)
            window_width, window_height = glfw.get_window_size(window)
            point_x = (xpos / window_width) * 2 - 1
            point_y = 1 - (ypos / window_height) * 2
            points.append((point_x, point_y))  # Add point to the list
but do not draw

        if key == glfw.KEY_E:  # Exit the program
            glfw.set_window_should_close(window, True)

# Mouse callback
def mouse_button_callback(window, button, action, mods):
    global squarePosX, squarePosY
    if button == glfw.MOUSE_BUTTON_LEFT and action == glfw.PRESS:
        xpos, ypos = glfw.get_cursor_pos(window)
        window_width, window_height = glfw.get_window_size(window)
        squarePosX = (xpos / window_width) * 2 - 1
        squarePosY = 1 - (ypos / window_height) * 2

def main():
```

```
    # Initialize the library
    if not glfw.init():
        return

    # Create a windowed mode window and its OpenGL context
    window = glfw.create_window(500, 500, "GLFW OpenGL", None, None)
    if not window:
        glfw.terminate()
        return

    # Make the window's context current
    glfw.make_context_current(window)

    # Set callbacks
    glfw.set_key_callback(window, key_callback)
    glfw.set_mouse_button_callback(window, mouse_button_callback)

    # Set up the rendering context
    glClearColor(0.0, 0.0, 0.0, 1.0)  # Black background
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0)

    # Loop until the user closes the window
    while not glfw.window_should_close(window):
        # Render here
        draw_square()

        # Swap front and back buffers
        glfw.swap_buffers(window)

        # Poll for and process events
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()
```

Output(s) (Screen Shot):


GLFW OpenGL


GLFW OpenGL



Conclusion and discussion:

Successfully understood the use of keyboard and mouse interface.

Date:

Signature of faculty in-charge

Post lab Question

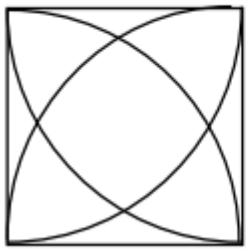
Write a program to draw the following

```
import turtle
polygon = turtle.Turtle()
```

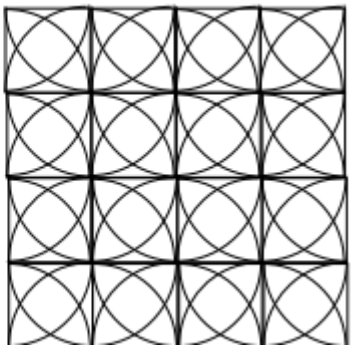
```
for i in range(4):
    polygon.forward(150)
    polygon.right(270)
    polygon.circle(150, 90)
    polygon.right(270)
turtle.done()
```

And

).



b).



I

```
import turtle
polygon = turtle.Turtle()
for k in range(2):
    for j in range(4):
        for i in range(4):
            polygon.forward(50)
            polygon.right(270)
            polygon.circle(50, 90)
            polygon.right(270)
        polygon.right(270)
    polygon.forward(50)
for l in range(2):
    for j in range(4):
        for i in range(4):
            polygon.forward(50)
            polygon.left(90)
            polygon.circle(50, 90)
            polygon.left(90)
        polygon.left(90)
    polygon.right(90)
    polygon.forward(50)
polygon.forward(50)
for l in range(2):
    for j in range(4):
        for i in range(4):
            polygon.forward(50)
```

```
            polygon.left(90)
            polygon.circle(50, 90)
            polygon.left(90)
        polygon.left(90)
    polygon.right(90)
    polygon.forward(50)
turtle.done()
```