#TASK\_01

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import random

import math

bg\_color=[1.0,1.0,1.0] # white background

rain\_color= [0.0, 0.0, 0.0] # rain color black

rain\_angle=90

def draw\_points(x,y):

glPointSize(7)

glBegin(GL\_POINTS)

if (bg\_color[0]==1.0 and bg\_color[1]==1.0 and bg\_color[2]==1.0) :

glColor3f(1.0,0.0, 0.0) #red

else:

glColor3f(0.545,0.0,0.0)

glVertex2f(x,y)

glEnd()

def rain\_droplets(x,y):

glLineWidth(2)

glBegin(GL\_LINES)

number\_of\_layers = 8 # Number of layers of rain

number\_of\_droplets = 50 # Number of raindrops in each layer

for i in range(number\_of\_layers):

for j in range(number\_of\_droplets):

x = random.randint(1, 1000)

length = 20

y = 900 - (i \* 100) # Adjust y position based on layer

x1 = x - length \* math.cos(math.radians(rain\_angle)) # Rain falls vertically

y1 = y - length

if rain\_color[0] >= 0.5:

glColor3f(1.0, 1.0, 1.0)

else:

glColor(\*rain\_color)

glVertex2f(x, y)

glVertex2f(x1, y1)

glEnd()

def draw\_triangle() :

glBegin(GL\_TRIANGLES)

if (bg\_color[0]==1.0 and bg\_color[1]==1.0 and bg\_color[2]==1.0) :

glColor3f(1.0,0.0, 0.0) #red

else:

glColor3f(0.545,0.0,0.0)

glVertex2f(300, 400)

glVertex2f(650, 400)

glVertex2f(470, 550)

glEnd()

def draw\_lines() :

glLineWidth(3)

glBegin(GL\_LINES)

if (bg\_color[0]==1.0 and bg\_color[1]==1.0 and bg\_color[2]==1.0) :

glColor3f(1.0,0.0, 0.0) #red

else:

glColor3f(0.545,0.0,0.0)

#house Frame

glVertex2f(310, 400)

glVertex2f(310, 150)

glVertex2f(640, 400)

glVertex2f(640, 150)

glVertex2f(310, 150)

glVertex2f(640, 150)

#window

glVertex2f(480, 370)

glVertex2f(620, 370)

glVertex2f(480, 370)

glVertex2f(480, 305)

glVertex2f(620, 370)

glVertex2f(620, 305)

glVertex2f(480, 305)

glVertex2f(620, 305)

glVertex2f(550, 305)

glVertex2f(550, 370)

glVertex2f(480, 337.5)

glVertex2f(620, 337.5)

#door geometry

glVertex2f(340, 150)

glVertex2f(340, 275)

glVertex2f(400, 150)

glVertex2f(400, 275)

glVertex2f(340, 275)

glVertex2f(400, 275)

glEnd()

def iterate():

glViewport(0, 0, 1000, 1000)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(0.0, 1000, 0.0, 1000, 0.0, 1.0)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

def background():

glClearColor(\*bg\_color, 1.0) # Set background color

glClear(GL\_COLOR\_BUFFER\_BIT)

def specialKeyListener(key, x, y):

global rain\_angle

if key == GLUT\_KEY\_LEFT:

rain\_angle -= 10 # Adjust rain angle left by 10 degrees

print("Rain bent towards left")

elif key == GLUT\_KEY\_RIGHT:

rain\_angle += 10 # Adjust rain angle right by 10 degrees

print("Rain bent towards right")

glutPostRedisplay()

def keyboardListener(key,x,y):

global bg\_color, rain\_color

if key==b'l':

if (bg\_color[0]>=1.0 and bg\_color[1]>=1.0 and bg\_color[2]>=1.0) and (rain\_color[0]<=0.0 and rain\_color[1]<=0.0 and rain\_color[2]<=0.0):

bg\_color = [1.0, 1.0, 1.0] # White bg color

rain\_color = [0.0, 0.0, 0.0] #Black rain

print("You are at the maximum brightness")

else:

bg\_color[0]+=0.5

bg\_color[1]+=0.5

bg\_color[2]+=0.5

rain\_color[0]-=0.5

rain\_color[1]-=0.5

rain\_color[2]-=0.5

print("Going to Light Mode")

if key==b'n':

if (bg\_color[0]<=0.0 and bg\_color[1]<=0.0 and bg\_color[2]<=0.0) and (rain\_color[0]>=1.0 and rain\_color[1]>=1.0 and rain\_color[2]>=1.0):

bg\_color = [0.0, 0.0, 0.0] # Black bg

rain\_color = [1.0, 1.0, 1.0] # White rain color

print("You are at the lowest brightness")

else:

bg\_color[0]-=0.5

bg\_color[1]-=0.5

bg\_color[2]-=0.5

rain\_color[0]+=0.5

rain\_color[1]+=0.5

rain\_color[2]+=0.5

print("Going to Dark Mode")

def showScreen():

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

glLoadIdentity()

iterate()

background()

draw\_lines()

draw\_triangle()

draw\_points(385, 210)

rain\_droplets(500, 10)

glutSwapBuffers()

def rain\_timer(value):

glutTimerFunc(100, rain\_timer, 0)

glutPostRedisplay()

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(1000, 1000)

glutInitWindowPosition(0, 0)

wind = glutCreateWindow(b"Mayeesha Mahbub 21201462 Lab01 Task01")

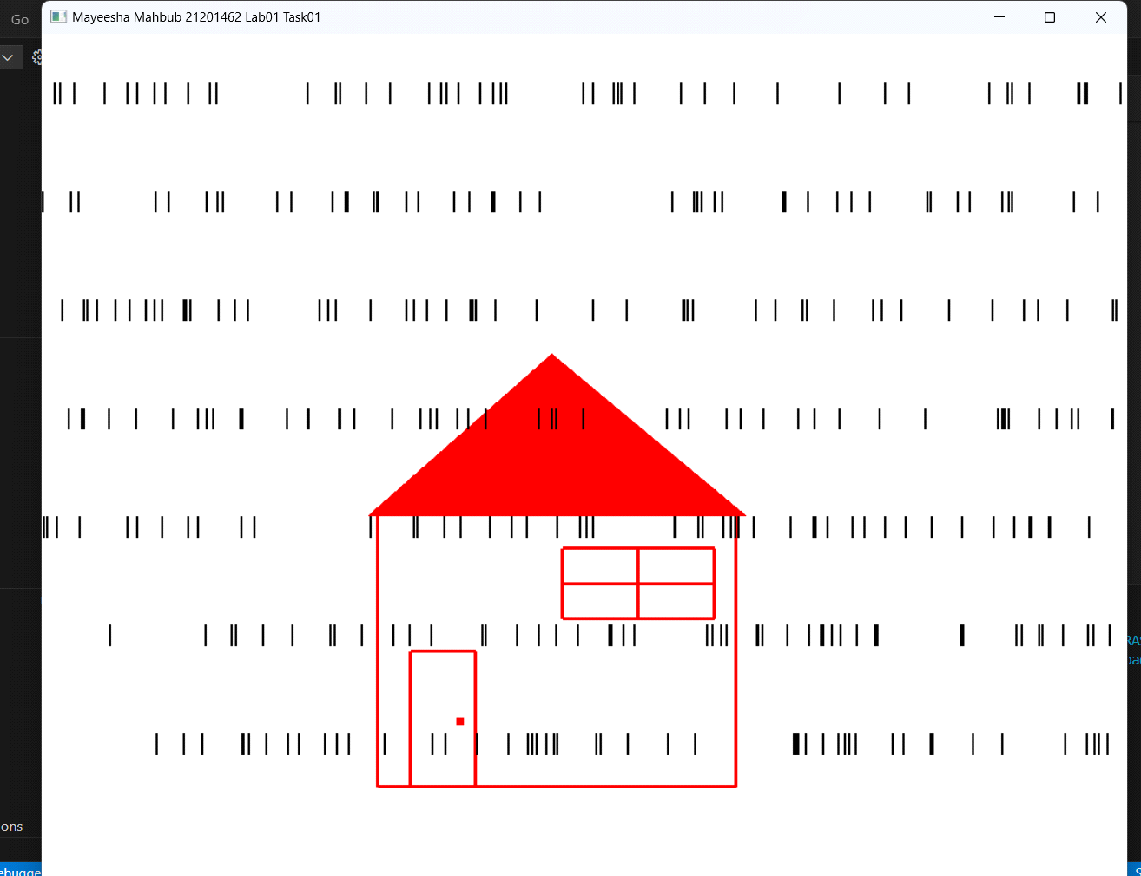
glutDisplayFunc(showScreen)

glutSpecialFunc(specialKeyListener)

glutKeyboardFunc(keyboardListener)

glutTimerFunc(100, rain\_timer, 0) # Start rain timer

glutMainLoop()



#TASK\_02

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import random

point = []

box\_left\_position = 0

box\_width = 900

box\_bottom\_position = 0

box\_height = 700

point\_size = 14

point\_speed = 0.05

clicked\_left\_mouse = False

clicked\_left\_mouse1 = None

initial\_time = 0

spacebar = False

colors = []

def draw\_point():

global clicked\_left\_mouse, initial\_time, spacebar

glEnable(GL\_POINT\_SMOOTH)

glPointSize(point\_size)

glBegin(GL\_POINTS)

for i in range(len(point)):

x, y, color, direction\_x, direction\_y = point[i]

if spacebar:

direction\_x = 0

direction\_y = 0

clicked\_left\_mouse = False

if clicked\_left\_mouse and not spacebar:

current\_time = glutGet(GLUT\_ELAPSED\_TIME)

time\_diff = (current\_time - initial\_time) % 900

if time\_diff < 100:

color = (0.0, 0.0, 0.0)

else:

original\_color = colors[i]

color = original\_color

glColor3f(\*color)

glVertex2f(x, y)

# Updating point position based on direction and speed

x += point\_speed \* direction\_x

y += point\_speed \* direction\_y

# Reflect points when they hit the borders

if x < box\_left\_position + point\_size:

x = box\_left\_position + point\_size

direction\_x = -direction\_x

if x > box\_width - point\_size:

x = box\_width - point\_size

direction\_x = -direction\_x

if y < box\_bottom\_position + point\_size:

y = box\_bottom\_position + point\_size

direction\_y = -direction\_y

if y > box\_height - point\_size:

y = box\_height - point\_size

direction\_y = -direction\_y

point[i] = (x, y, color, direction\_x, direction\_y)

glEnd()

def create\_random\_point(x, y):

if box\_left\_position < x < box\_width and box\_bottom\_position < y < box\_height:

# Random colors for generated points

r, g, b = (random.random(), random.random(), random.random())

color = (r, g, b)

# Random direction in diagonals

direction\_x = random.choice([-1, 1])

direction\_y = random.choice([-1, 1])

point.append((x, y, color, direction\_x, direction\_y))

colors.append(color)

def mouseListener(button, state, x, y):

global clicked\_left\_mouse, initial\_time, spacebar

if spacebar:

return

if button == GLUT\_RIGHT\_BUTTON and state == GLUT\_DOWN:

create\_random\_point(x,700 - y)

print("New point added")

if button == GLUT\_LEFT\_BUTTON and state == GLUT\_DOWN:

if clicked\_left\_mouse == False:

clicked\_left\_mouse = not clicked\_left\_mouse

print("Blinking Function Turned On")

else:

clicked\_left\_mouse = not clicked\_left\_mouse

print("Blinking Function Turned Off")

def specialKeysListener(key, x, y):

global point\_speed, clicked\_left\_mouse, spacebar

if key == GLUT\_KEY\_UP:

point\_speed += 0.005

print("Speed Increased")

elif key == GLUT\_KEY\_DOWN:

if point\_speed<=0:

point\_speed=0

print("Speed limit reached")

else:

point\_speed -= 0.005

print("Speed Decreased")

def keyboardListener(key, x, y):

global spacebar, point\_speed, clicked\_left\_mouse, clicked\_left\_mouse1

if key == b" ":

spacebar = not spacebar

if spacebar:

clicked\_left\_mouse1 = clicked\_left\_mouse

clicked\_left\_mouse = False

for i in range(len(point)):

x, y, color, direction\_x, direction\_y = point[i]

direction\_x = 0

direction\_y = 0

point[i] = (x, y, color, direction\_x, direction\_y)

print("Point Staus: Freeze")

else:

clicked\_left\_mouse = clicked\_left\_mouse1

clicked\_left\_mouse1 = None

for i in range(len(point)):

direction\_x = random.choice([-1, 1])

direction\_y = random.choice([-1, 1])

point[i] = (point[i][0], point[i][1], point[i][2], direction\_x, direction\_y)

print("Point Staus: Moving")

def draw\_box():

glLineWidth(2)

glBegin(GL\_LINES)

glVertex2f(box\_width, box\_height)

glVertex2f(box\_left\_position, box\_height)

glVertex2f(box\_left\_position, box\_height)

glVertex2f(box\_left\_position, box\_bottom\_position)

glVertex2f(box\_width, box\_bottom\_position)

glVertex2f(box\_left\_position, box\_bottom\_position)

glVertex2f(box\_width, box\_bottom\_position)

glVertex2f(box\_width, box\_height)

glEnd()

def iterate():

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(0, 900, 0, 700, -1, 1)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

def display():

glClear(GL\_COLOR\_BUFFER\_BIT)

glLoadIdentity()

iterate()

glColor3f(1.0, 1.0, 1.0)

draw\_box()

draw\_point()

glutSwapBuffers()

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(900, 700)

glutCreateWindow(b"21201462 Mayeesha Mahbub Lab01 Task02")

glutDisplayFunc(display)

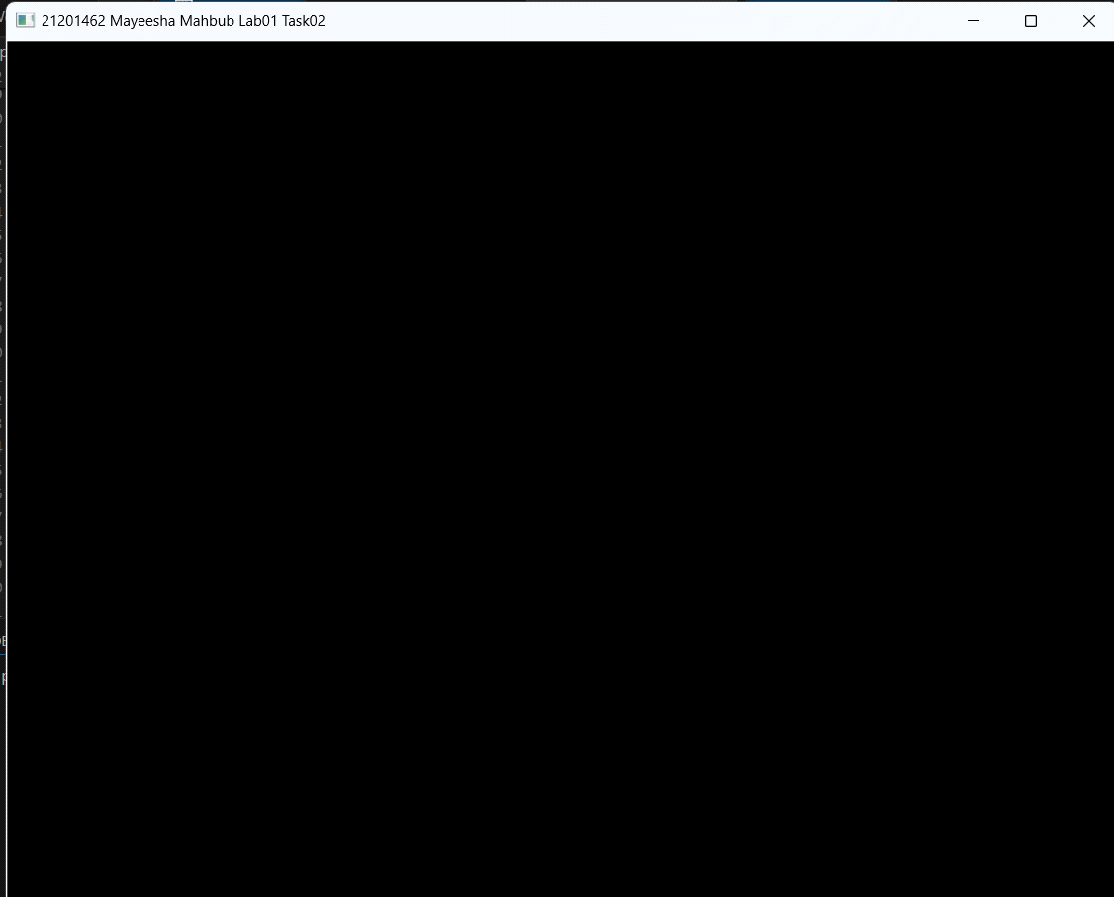
glutMouseFunc(mouseListener)

glutSpecialFunc(specialKeysListener)

glutKeyboardFunc(keyboardListener)

glutIdleFunc(display)

glutMainLoop()



# I couldn't finished the task 2 and having multiple errors .