**Task 1: Building a House in Rainfall**

**Code:**

#!/usr/bin/env python3

#####################################################

######## CSE423\_Lab01\_Task01 ####################

######## Abir Ahammed Bhuiyan ##################

######## 20101197 ################

######## section: 01 ##############

##########################################

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import random

class Rain:

def \_\_init\_\_(self, x1, y1, x2, y2):

self.x1 = x1

self.y1 = y1

self.x2 = x2

self.y2 = y2

class RainScene:

def \_\_init\_\_(self):

self.width = self.height = 500

self.rain\_size = 2

self.rain\_drop\_list = []

self.rain\_speed = 0.05

self.rain\_density = 0.009

self.rain\_angle = 0

self.background = 0

def drawHouse(self):

glBegin(GL\_QUADS)

glColor3f(0.44, 0.22, 0.22)

glVertex2f(300, 0)

glVertex2f(150, 0)

glVertex2f(150, 130)

glVertex2f(300, 130)

glEnd()

glBegin(GL\_TRIANGLES)

glColor3f(0.47, 0.75, 0.46)

glVertex2f(330, 130)

glVertex2f(120, 130)

glVertex2f(225, 250)

glEnd()

glBegin(GL\_QUADS)

glColor3f(0.48, 0.13, 0.6)

glVertex2f(250, 35)

glVertex2f(200, 35)

glVertex2f(200, 84)

glVertex2f(250, 84)

glEnd()

def rainMaker(self):

delay = random.random()

if delay < self.rain\_density:

x1 = random.randint(-800, self.width+800)

y1 = self.height

x2 = x1

y2 = self.height+random.randint(20, 70)

self.rain\_drop\_list.append(Rain(x1, y1, x2, y2))

def drawRain(self):

for rain in self.rain\_drop\_list:

glPushMatrix()

glTranslatef((rain.x1 + rain.x2)/2, (rain.y1 + rain.y2)/2, 0)

glRotatef(self.rain\_angle, 0, 0, 1)

glTranslatef(-(rain.x1 + rain.x2)/2, -(rain.y1 + rain.y2)/2, 0)

glLineWidth(self.rain\_size)

glBegin(GL\_LINES)

glColor3f(0.39, 0.4, 1)

glVertex2f(rain.x1, rain.y1)

glVertex2f(rain.x2, rain.y2)

glEnd()

glPopMatrix()

def iterate(self):

glViewport(0, 0, self.width, self.height)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(0.0, self.width, 0.0, self.height, 0.0, 1.0)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

def showScreen(self):

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

glLoadIdentity()

self.iterate()

glClearColor(self.background, self.background, self.background, 1.0)

self.rainMaker()

self.drawRain()

self.drawHouse()

glutSwapBuffers()

def animate(self):

for idx, rain in enumerate(self.rain\_drop\_list):

if rain.y1 >= 0 and rain.y1 <= self.height:

if self.rain\_angle < -5 and self.rain\_angle >= -45:

rain.x1 -= self.rain\_speed

rain.x2 -= self.rain\_speed

rain.y1 -= self.rain\_speed

rain.y2 -= self.rain\_speed

if self.rain\_angle > 5 and self.rain\_angle <= 45:

rain.x1 += self.rain\_speed

rain.x2 += self.rain\_speed

rain.y1 -= self.rain\_speed

rain.y2 -= self.rain\_speed

if self.rain\_angle > -5 and self.rain\_angle < 5:

rain.y1 -= self.rain\_speed

rain.y2 -= self.rain\_speed

else:

self.rain\_drop\_list.pop(idx)

glutPostRedisplay()

def keyboardListener(self, key, x, y):

if key == b"=": # + key

self.rain\_density += 0.001

if key == b"-":

self.rain\_density -= 0.001

if key == b"f":

self.rain\_speed += 0.01

if key == b"s":

self.rain\_speed -= 0.01

glutPostRedisplay()

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

pass

def specialKeyListener(self, key, x, y):

if key == GLUT\_KEY\_RIGHT:

if not self.rain\_angle >= 45:

self.rain\_angle += 1

if key == GLUT\_KEY\_LEFT:

if not self.rain\_angle <= -45:

self.rain\_angle -= 1

if key == GLUT\_KEY\_UP:

if not self.background > 1:

self.background += 0.05

if key == GLUT\_KEY\_DOWN:

if not self.background < 0:

self.background -= 0.05

glutPostRedisplay()

def run(self):

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(self.width, self.height)

glutInitWindowPosition(0, 0)

glutCreateWindow("OpenGL Rain Drop")

glutDisplayFunc(self.showScreen)

glutIdleFunc(self.animate)

glutKeyboardFunc(self.keyboardListener)

glutMouseFunc(self.mouseListener)

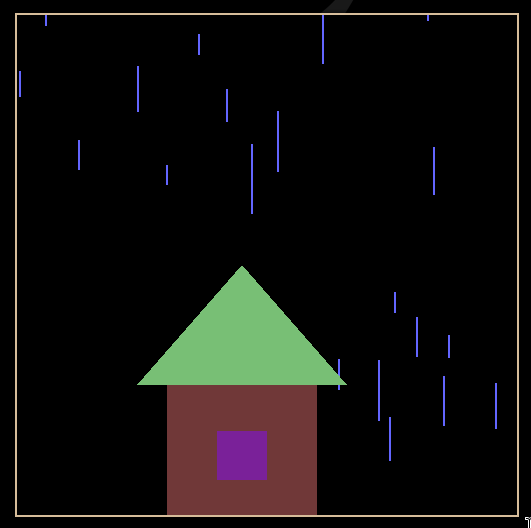
glutSpecialFunc(self.specialKeyListener)

glutMainLoop()

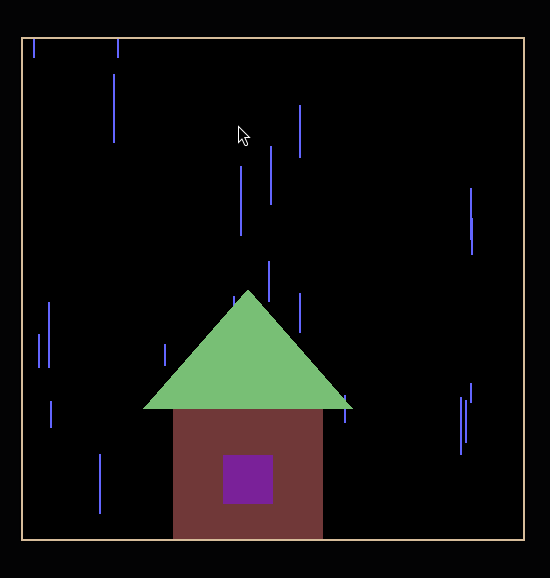
if \_\_name\_\_ == "\_\_main\_\_":

RainScene().run()

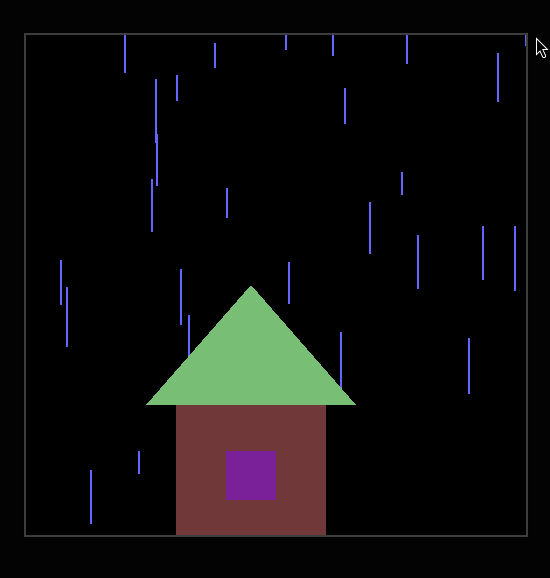
**Sub\_task (i):**

****

**Sub\_task (ii):** right/left arrow -> bend

****

**Sub\_task (iii):** up/down arrow -> change background



**Task 2: Building the Amazing Box**

**Code:**

#!/usr/bin/env python3

#####################################################

######## CSE423\_Lab01\_Task02 ####################

######## Abir Ahammed Bhuiyan ##################

######## 20101197 ################

######## section: 01 ##############

##########################################

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import random

class Point:

def \_\_init\_\_(self, x, y):

self.size = 10

self.r = random.random()

self.g = random.random()

self.b = random.random()

self.x = x

self.y = y

self.direction\_x = int(random.sample([1, -1], 1)[0])

self.direction\_y = int(random.sample([1, -1], 1)[0])

def \_\_str\_\_(self):

return f"RGB -> ({self.r}, {self.g}, {self.b})"

class PointScene:

def \_\_init\_\_(self):

self.width = self.height = 500

self.background = 0

self.speed = 0.001

self.point\_number = 5

self.point\_list = []

self.blink = False

self.freeze = False

self.timer = 0

def makePoints(self, x, y):

for \_ in range(self.point\_number):

self.point\_list.append(Point(x, self.height-y))

def drawPoints(self):

for point in self.point\_list:

glPointSize(point.size)

glBegin(GL\_POINTS)

if self.blink and not self.freeze:

if not random.randint(0, 50)%2:

glColor3f(point.r, point.g, point.b)

else:

glColor3f(self.background, self.background, self.background)

self.timer += 1

else:

glColor3f(point.r, point.g, point.b)

glVertex2f(point.x, point.y)

glEnd()

def iterate(self):

glViewport(0, 0, self.width, self.height)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(0.0, self.width, 0.0, self.height, 0.0, 1.0)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

def showScreen(self):

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

glLoadIdentity()

self.iterate()

glClearColor(self.background, self.background, self.background, 1.0)

self.drawPoints()

glutSwapBuffers()

def animate(self):

if not self.freeze:

for point in self.point\_list:

point.x += point.direction\_x \* self.speed

point.y += point.direction\_y \* self.speed

glutPostRedisplay()

def keyboardListener(self, key, x, y):

if key == b' ':

self.freeze = False if self.freeze else True

glutPostRedisplay()

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

if not self.freeze:

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

self.makePoints(x, y)

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

self.blink = False if self.blink else True

glutPostRedisplay()

def specialKeyListener(self, key, x, y):

if not self.freeze:

if key == GLUT\_KEY\_UP:

self.speed += 0.001

if key == GLUT\_KEY\_DOWN:

self.speed -= 0.001

glutPostRedisplay()

def run(self):

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(self.width, self.height)

glutInitWindowPosition(0, 0)

glutCreateWindow("OpenGL Point Spawn")

glutDisplayFunc(self.showScreen)

glutIdleFunc(self.animate)

glutKeyboardFunc(self.keyboardListener)

glutMouseFunc(self.mouseListener)

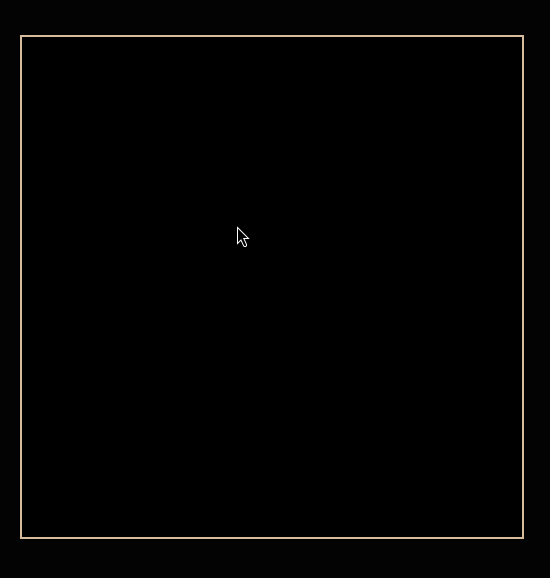
glutSpecialFunc(self.specialKeyListener)

glutMainLoop()

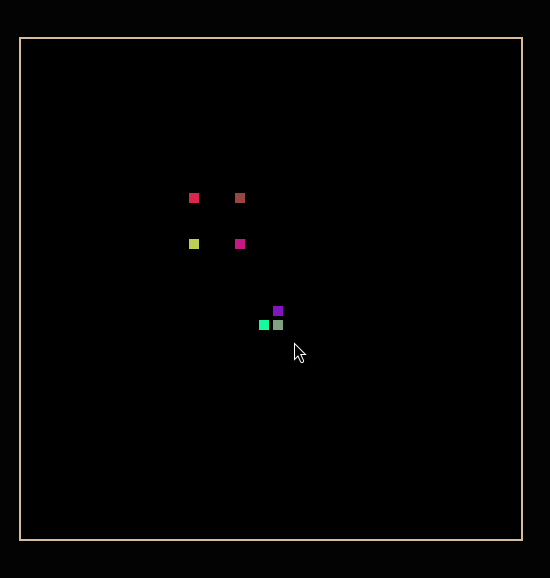
if \_\_name\_\_ == "\_\_main\_\_":

PointScene().run()

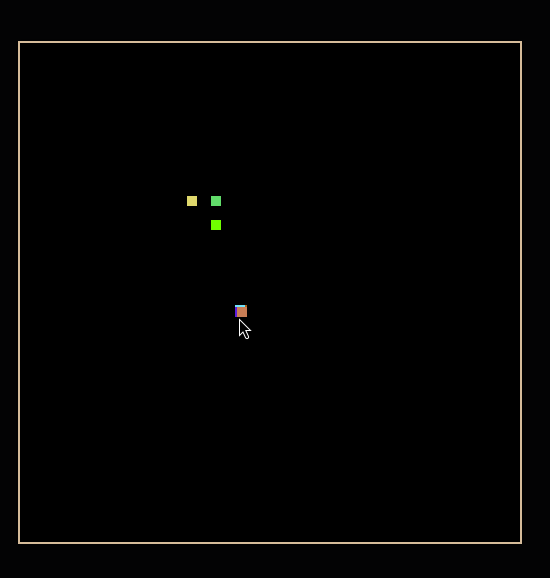
**Sub\_task (i):** right button click -> generate point



**Sub\_task (ii):** up arrow -> speed increased; down arrow -> speed decreased



**Sub\_task (iii):** left button click -> blink



**Sub\_task (iv):** space button -> freeze/unfreeze

