Task:1

```
import random
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *


direction=0


bg=0
r=1.0
g=1.0
b=1.0


def house():
    glPointSize(10)
    glColor3f(r,g,b)

    glBegin(GL_TRIANGLES) #ceeling/roof
    glVertex2f(0,100)
    glVertex2d(90,0)
    glVertex2d(-90,0)
    glEnd()



    glBegin(GL_LINES) #pillars
    glColor3f(r,g,b)

    glVertex2f(71,0) #rightwall
    glVertex2f(71,-120)
```

```
glVertex2f(-71,0) #leftwall
glVertex2f(-71,-120)

glVertex2f(71,-120) #bottom floor
glVertex2f(-71,-120)

glEnd()

glBegin(GL_LINES) #enterance
glColor3f(r,g,b)

glVertex2f(-50,-50) #left
glVertex2f(-50,-120)

glVertex2f(-10,-50) #right
glVertex2f(-10,-120)

glVertex2f(-50,-50) #bottom floor
glVertex2f(-10,-50)

glEnd()

glBegin(GL_POINTS) #door knob
glVertex2f(-20,-90)
glEnd()


glBegin(GL_LINES) #window
glColor3f(r,g,b)

glVertex2f(50,-30) #right
```

```
    glVertex2f(50,-50)

    glVertex2f(10,-30) #left
    glVertex2f(10,-50)

    glVertex2f(10,-30) #top
    glVertex2f(50,-30)

    glVertex2f(10,-50) #bottom
    glVertex2f(50,-50)

    glVertex2f(30,-30) #Crosshair
    glVertex2f(30,-50)

    glVertex2f(10,-40) #Crosshair
    glVertex2f(50,-40)
    glEnd()




def rain():
    global direction
    glLineWidth(3)
    glColor3f(0.1,0.8,1)

    glBegin(GL_LINES)
    for i in range(400):
```

```
        x = random.uniform(-500, 500)
        y = random.uniform(-100, 500)
        if -90<x<90:
            y = random.uniform(0, 500)


        length = random.uniform(5,10)

        glVertex2f(x, y)
        glVertex2f(x+direction, y+length)
    glEnd()


def keyboardListener(key,x,y):
    global bg
    global r
    global g
    global b
    global direction

    if key==b'a':
        bg+=0.05
        r-=0.05
        g-=0.05
        b-=0.05

        print("Night to day !")

    if key==b'f':
        bg-=0.05
```

```
        r+=0.05
        g+=0.05
        b+=0.05

        print("Day to night !")

    if key==b'r':
        bg=0
        r=1.0
        g=1.0
        b=1.0
        direction=0

        print("Reset Done !")

    glutPostRedisplay()




def specialKeyListener(key,x,y):
    global direction

    if key==GLUT_KEY_RIGHT:
        direction-=1
        print("Going Right")

    if key==GLUT_KEY_LEFT:
        direction+=1
        print("Going Left")
```

```
def display():
    #//clear the display
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glClearColor(bg,bg,bg,0);   #//color black
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    #//load the correct matrix -- MODEL-VIEW matrix
    glMatrixMode(GL_MODELVIEW)
    #//initialize the matrix
    glLoadIdentity()
    #//now give three info
    #//1. where is the camera (viewer)?
    #//2. where is the camera looking?
    #//3. Which direction is the camera's UP direction?
    gluLookAt(0,0,200,  0,0,0,  0,1,0)
    glMatrixMode(GL_MODELVIEW)

    rain()
    house()
    glutSwapBuffers()

def animate():
    #//codes for any changes in Models, Camera
    glutPostRedisplay()

def init():
    #//clear the screen
    glClearColor(0,0,0,0)
    #//load the PROJECTION matrix
    glMatrixMode(GL_PROJECTION)
    #//initialize the matrix
```

```
    glLoadIdentity()
    #//give PERSPECTIVE parameters
    gluPerspective(104, 1,  1,  1000.0)
    # **(important)**aspect ratio that determines the field of view in the X direction (horizontally). The bigger this angle is, the more you can see of the world - but at the same time, the objects you can see will become smaller.
    #//near distance
    #//far distance


glutInit()
glutInitWindowSize(1000, 1000)
glutInitWindowPosition(0, 0)
glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB) #  //Depth, Double buffer, RGB color

# glutCreateWindow("My OpenGL Program")
wind = glutCreateWindow(b"Task-1 House in the rain")
init()

glutDisplayFunc(display)    #display callback function
glutIdleFunc(animate)   #what you want to do in the idle time (when no drawing is occuring)

glutKeyboardFunc(keyboardListener)
glutSpecialFunc(specialKeyListener)

glutMainLoop()      #The main loop of OpenGL
```

Task:2

```
import random
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
```

```
data=[]
box_left=0
box_low=0
height=500
width=1000
init_speed=0.04
init_left_mouse=False
left_mouse=None
space=False
color_data=[]

def random_points(x,y):
    if box_left<x<width and box_low<y<height:
        color_data.append([random.random(),random.random(),random.random()])
        x_dir=random.choice([-1,1])
        y_dir=random.choice([-1,1])
        data.append([x,y,color_data[-1],x_dir,y_dir])



def points():
    global init_left_mouse,space,interval
    glPointSize(10)
    glBegin(GL_POINTS)

    for i in range (len(data)):
        x,y,color,dir_x,dir_y=data[i]
```

```
if space==True:
    dir_x=0
    dir_y=0
    init_left_mouse=False

if init_left_mouse==True and space==False:
    time=glutGet(GLUT_ELAPSED_TIME)//1000
    if time%2==0:
        glColor3f(0,0,0)
    else:
        glColor3f(color[0],color[1],color[2])
else:
    glColor3f(color[0],color[1],color[2])



glVertex2f(x,y)

x+=dir_x*init_speed  #updating the direction
y+=dir_y*init_speed

#Boundary corner case
if x<box_left+10:
    x=box_left+10
    dir_x=-dir_x

if x>width-10:
    x=width-10
    dir_x=-dir_x
```

```
        if y<box_low+10:
            y=box_low+10
            dir_y=-dir_y

        if y>height-10:
            y=height-10
            dir_y=-dir_y
        data[i]=[x,y,color,dir_x,dir_y]
    glEnd()




def mouseListener(button,state,x,y):
    global space,height,init_left_mouse
    if space==True:
        return      #If the spacebar is true the function will exit here

    if button==GLUT_LEFT_BUTTON and state==GLUT_DOWN:
        if init_left_mouse==False:
            init_left_mouse= True
            print("Blinking started")

        elif init_left_mouse==True:
            init_left_mouse=False
            print("Blinking stopped")
    if button==GLUT_RIGHT_BUTTON and state==GLUT_DOWN:
```

```
        random_points(x,height-y)
        print("Random points added")
    glutPostRedisplay()




def specialKeyListener(key, x, y):
    global init_speed

    if key==GLUT_KEY_DOWN:
        if init_speed==0 or init_speed<0:
            print("This is the lowest speed!")
            init_speed=0
        else:
                init_speed-=0.01
                print("Lowering Speed")

    if key==GLUT_KEY_UP:
        init_speed+=0.01
        print("Increasing speed")

    glutPostRedisplay()

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

```
if key == b' ':
    global space
    space = not space
    if space == True:
        global init_left_mouse,left_mouse
        left_mouse=init_left_mouse
        left_mouse=False
        for i in range (len(data)):
            x,y,color,dir_x,dir_y=data[i]
            new_x=0
            new_y=0
            data[i]=[x,y,color,new_x,new_y]
        print("Paused")



    else:
        init_left_mouse=left_mouse
        left_mouse=None
        for i in range (len(data)):
            x,y,color,dir_x,dir_y=data[i]


            new_x=random.choice([-1,1])
            new_y=random.choice([-1,1])
            data[i]=[x,y,color,new_x,new_y]
        print("Resumed")
elif key == b'r':
    global init_speed
    init_speed=0.04
    init_left_mouse=False
```

```
        left_mouse=None

    glutPostRedisplay()



#Driver code from here
def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, width, 0, height)  # Use orthographic projection for 2D

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # Draw the initial box
    glLineWidth(1)
    glBegin(GL_LINES)

    glVertex2f(width, height)     # Upper Boundary
    glVertex2f(box_left, height)

    glVertex2f(box_left, height)  # Left Boundary
    glVertex2f(box_left, box_low)

    glVertex2f(width, box_low)    # Lower Boundary
    glVertex2f(box_left, box_low)
```

```
    glVertex2f(width, box_low)    # Right Boundary
    glVertex2f(width, height)

    glEnd()
    points()

    glutSwapBuffers()

def animate():
    glutPostRedisplay()

def init():
    glClearColor(0, 0, 0, 0)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, width, 0, height)  # Set orthographic projection for 2D
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitWindowSize(width, height)
glutInitWindowPosition(10, 10)
glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB) # Depth, Double buffer, RGB color

wind = glutCreateWindow(b"Random Shooter Balls")
init()

glutDisplayFunc(display)  # Display callback function
```

```
glutIdleFunc(animate)    # Update function for animation
glutKeyboardFunc(keyboardListener)
glutSpecialFunc(specialKeyListener)
glutMouseFunc(mouseListener)
glutMainLoop()  # The main loop of OpenGL
```