Name: Nowshin Sumaiya

ID: 21301276

Sec:06

Course:CSE423 (lab02)

```
import random
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
W_Width, W_Height = 500, 500

class Catcher:
    def __init__(self, x=0, y=-230, length=170, width=18) :
        self.position = {'x': x, 'y': y}
        self.length = length
        self.width = width
        self.color = (1.0, 1.0, 1.0)
    def draw(self):
        glColor3f(*self.color)
        glLineWidth(3)
        half_length = self.length / 2
        half_width = self.width / 2
        glBegin(GL_LINES)

        self.draw_midpoint_line(int(self.position['x'] + half_width), int(self.position['y'] -
half_width),
                                int(self.position['x'] - half_width), int(self.position['y'] - half_width))
        self.draw_midpoint_line(int(self.position['x'] - half_width), int(self.position['y'] -
half_width),
                                int(self.position['x'] - half_length), int(self.position['y'] + half_width))
        self.draw_midpoint_line(int(self.position['x'] - half_length), int(self.position['y'] +
half_width),
                                int(self.position['x'] + half_length), int(self.position['y'] + half_width))
```

```
        self.draw_midpoint_line(int(self.position['x'] + half_length), int(self.position['y'] +
half_width),
                                int(self.position['x'] + half_width), int(self.position['y'] - half_width))
        glEnd()

    @staticmethod
    def draw_midpoint_line(x1, y1, x2, y2):
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        x, y = x1, y1
        step_x = 1 if x1 < x2 else -1
        step_y = 1 if y1 < y2 else -1
        if dx > dy  :
            p = 2 * dy - dx
            for _ in range(dx + 1):
                glVertex2f(x, y)
                if p >= 0:
                    y += step_y
                    p -= 2 * dx
                x += step_x
                p += 2 * dy
        else:
            p = 2 * dx - dy
            for _ in range(dy + 1):
                glVertex2f(x, y)
                if p >= 0:
                    x += step_x
                    p -= 2 * dy
                y += step_y
                p += 2 * dx
#------------------------------------------------------------
class RedX:
    def __init__(self):
        self.position = {'x': 200, 'y': 200}
        self.size = 40
        self.color = (1.0, 0.0, 0.0)
    def draw(self):
        glColor3f(*self.color)
        glLineWidth(5)
        half_size = self.size / 2
```

```
        glBegin(GL_LINES)

        self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y'] - half_size),
                                int(self.position['x'] + half_size), int(self.position['y'] + half_size))

        self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y'] + half_size),
                                int(self.position['x'] + half_size), int(self.position['y'] - half_size))
        glEnd()

    @staticmethod
    def draw_midpoint_line(x1, y1, x2, y2) :
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        x, y = x1, y1
        step_x = 1 if x1 < x2 else -1
        step_y = 1 if y1 < y2 else -1
        if dx > dy :
            p = 2 * dy - dx

            for _ in range(dx + 1) :
                glVertex2f(x, y)
                if p >= 0:
                    y += step_y
                    p -= 2 * dx
                x += step_x
                p += 2 * dy
        else:
            p = 2 * dx - dy
            for _ in range(dy + 1):
                if y != int((y1 + y2) / 2):
                    glVertex2f(x, y)
                if p >= 0:
                    x += step_x
                    p -= 2 * dy
                y += step_y
                p += 2 * dx
#-------------------------
class Diamond:
    def __init__(self, x=0, y=0, size=30):
        self.position = {'x': x, 'y': y}
```

```
    self.size = size
    self.color = (random.random(), random.random(), random.random() )
def draw(self):
    glColor3f(*self.color)
    glPointSize(2.0)
    half_size = self.size / 2
    glBegin(GL_POINTS)

    self.draw_midpoint_line(int(self.position['x']), int(self.position['y'] - half_size),
                            int(self.position['x'] + half_size), int(self.position['y']))
    self.draw_midpoint_line(int(self.position['x'] + half_size), int(self.position['y']),
                            int(self.position['x']), int(self.position['y'] + half_size))
    self.draw_midpoint_line(int(self.position['x']), int(self.position['y'] + half_size),
                            int(self.position['x'] - half_size), int(self.position['y']))
    self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y']),
                            int(self.position['x']), int(self.position['y'] - half_size))
    glEnd()

@staticmethod
def draw_midpoint_line(x1, y1, x2, y2):
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)
    x, y = x1, y1
    step_x = 1 if x1 < x2 else -1
    step_y = 1 if y1 < y2 else -1
    if dx > dy:
        p = 2 * dy - dx
        for _ in range(dx + 1):
            glVertex2f(x, y)
            if p >= 0:
                y += step_y
                p -= 2 * dx
            x =x+ step_x
            p += 2 * dy
    else:
        p = 2 * dx - dy
        for _ in range(dy + 1):
            glVertex2f(x, y)
            if p >= 0:
                x += step_x
```

```
                p -= 2 * dy
            y += step_y
            p += 2 * dx
#-----------------------------------------------
class Restart:
    def __init__(self):
        self.position = {'x': -200, 'y': 200}
        self.size = 60
        self.color = (0.0, 1.0, 1.0)
    def draw(self):
        glColor3f(*self.color)
        glLineWidth(7)
        half_size = self.size / 2
        glBegin(GL_LINES)

        self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y']),
                                int(self.position['x'] + half_size / 2), int(self.position['y']))
        self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y']),
                                int(self.position['x']), int(self.position['y'] - half_size / 2))
        self.draw_midpoint_line(int(self.position['x'] - half_size), int(self.position['y']),
                                int(self.position['x']), int(self.position['y'] + half_size / 2))
        glEnd()

    @staticmethod
    def draw_midpoint_line(x1, y1, x2, y2):
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        x, y = x1, y1
        step_x = 1 if x1 < x2 else -1
        step_y = 1 if y1 < y2 else -1
        if dx > dy:
            p = 2 * dy - dx
            for _ in range(dx + 1):
                glVertex2f(x, y)
                if p >= 0:
                    y += step_y
                    p -= 2 * dx
                x += step_x
                p += 2 * dy
        else:
```

```
            p = 2 * dx - dy
            for _ in range(dy + 1):
                glVertex2f(x, y)
                if p >= 0:
                    x += step_x
                    p -= 2 * dy
                y += step_y
                p += 2 * dx
restart_button = Restart()
#-----------------------------------------------
class PlayPause:
    def __init__(self):
        self.position = {'x': 0, 'y': 200}
        self.size = 40
        self.color = (1.0, 1.75, 0.0)
        self.paused = False
    def draw(self):
        glColor3f(*self.color)
        glLineWidth(3)
        half_size = self.size / 2
        glBegin(GL_LINES)
        if self.paused:

            self.draw_midpoint_line(int(self.position['x'] - half_size / 2 - 10), int(self.position['y'] +
half_size),
                                    int(self.position['x'] + half_size / 2), int(self.position['y']))
            self.draw_midpoint_line(int(self.position['x'] - half_size / 2 - 10), int(self.position['y'] -
half_size),
                                    int(self.position['x'] - half_size / 2 - 10), int(self.position['y'] + half_size))
            self.draw_midpoint_line(int(self.position['x'] - half_size / 2 - 10), int(self.position['y'] -
half_size),
                                    int(self.position['x'] + half_size / 2), int(self.position['y']))
        else:
            self.draw_midpoint_line(int(self.position['x'] - 10), int(self.position['y'] + half_size),
                                    int(self.position['x'] - 10), int(self.position['y'] - half_size))
            self.draw_midpoint_line(int(self.position['x'] + 10), int(self.position['y'] + half_size),
                                    int(self.position['x'] + 10), int(self.position['y'] - half_size))
        glEnd()

    @staticmethod
```

```
    def draw_midpoint_line(x1, y1, x2, y2):
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        x, y = x1, y1
        step_x = 1 if x1 < x2 else -1
        step_y = 1 if y1 < y2 else -1
        if dx > dy:
            p = 2 * dy - dx
            for _ in range(dx + 1):
                glVertex2f(x, y)
                if p >= 0:
                    y += step_y
                    p -= 2 * dx
                x += step_x
                p += 2 * dy
        else:
            p = 2 * dx - dy
            for _ in range(dy + 1):
                if y != int((y1 + y2) / 2):
                    glVertex2f(x, y)
                if p >= 0:
                    x += step_x
                    p -= 2 * dy
                y += step_y
                p += 2 * dx

play_pause_button = PlayPause()
#----------------------------------------------
def convert_coordinator(x, y):
    global W_Width, W_Height
    temp1 = x - (W_Width / 2)
    temp2 = (W_Height / 2) - y
    return temp1, temp2
def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glClearColor(0, 0, 0, 0)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt(0, 0, 200, 0, 0, 0, 0, 1, 0)
```

```
    glMatrixMode(GL_MODELVIEW)
    catcher.draw()
    diamond.draw()
    restart_button.draw()
    RedX().draw()
    play_pause_button.draw()

    glutSwapBuffers()
#-------------------------------------
score = 0
game_over = False
falling_speed = 1.5
speed_increment = 0.3

def animate(_):
    global diamond, catcher, score, game_over, falling_speed, play_pause_button
    if not play_pause_button.paused:
        if not game_over:
            diamond.position['y'] -= falling_speed
            if diamond.position['y'] - diamond.size < -250:
                game_over = True
                diamond.position['y'] = -500
                catcher.color = (1.0, 0.0, 0.0)
                print(f"Game Over!! final score: {score}")
            elif diamond.position['y'] - diamond.size < catcher.position['y'] + 10 and \
                    catcher.position['x'] - catcher.length / 2 < diamond.position['x'] < \
                    catcher.position['x'] + catcher.length / 2:
                diamond.position['y'] = 250
                diamond.position['x'] = random.uniform(-240, 240)
                diamond.color = (random.random(), random.random(), random.random())
                score = score+ 1
                falling_speed += speed_increment
                print(f"Score: {score}")

    glutPostRedisplay()
    glutTimerFunc(17, animate, 0)
#---------------------------------
def mouse(button, state, x, y):
    global game_over, score, falling_speed, play_pause_button
    if button == GLUT_LEFT_BUTTON and state == GLUT_DOWN:
```

```
        x, y = convert_coordinator(x, y)
        if -30 <= x <= 30 and 160 <= y <= 240:
            if play_pause_button.paused:
                play_pause_button.paused = False
                glutTimerFunc(15, animate, 0)
                print("Game Resumed")
            else:
                play_pause_button.paused = True
                print("Game Paused")
            glutPostRedisplay()
            return

        if -320 <= x <= -160 and 160 <= y <= 240:
            game_over = False
            score = 0
            falling_speed = 2.0
            catcher.color = (1.0, 1.0, 1.0)
            catcher.position = {"x": 0, "y": -240}
            diamond.position['y'] = 250
            diamond.position['x'] = random.uniform(-240, 240)
            diamond.color = (random.random(), random.random(), random.random())
            print("Game Restarted")
        if 140 <= x <= 240 and 180 <= y <= 250:
            print(f"Goodbye. your final score: {score}")
            glutLeaveMainLoop()
#---------------------------------------------------------------------
def specialKey(key, _, __):
    global catcher, play_pause_button
    if not play_pause_button.paused:
        if key == GLUT_KEY_LEFT:
            catcher.position['x'] -= 10
        if key == GLUT_KEY_RIGHT:
            catcher.position['x'] += 10
        catcher.position['x'] = max(-240, min(240, catcher.position['x']))
        glutPostRedisplay()
#---------------------------------------------------------------------
def init():
    glClearColor(0, 0, 0, 0)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
```

```
    gluPerspective(104, 1, 1, 1000.0)
    glutMouseFunc(mouse)
glutInit()
glutInitWindowSize(W_Width, W_Height)
glutInitWindowPosition(0, 0)
glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB)
glutCreateWindow(b'Assignment02 game')
init()
diamond = Diamond()
catcher = Catcher()
glutDisplayFunc(display)
glutTimerFunc(25, animate, 0)
glutSpecialFunc(specialKey)
glutMainLoop()
```

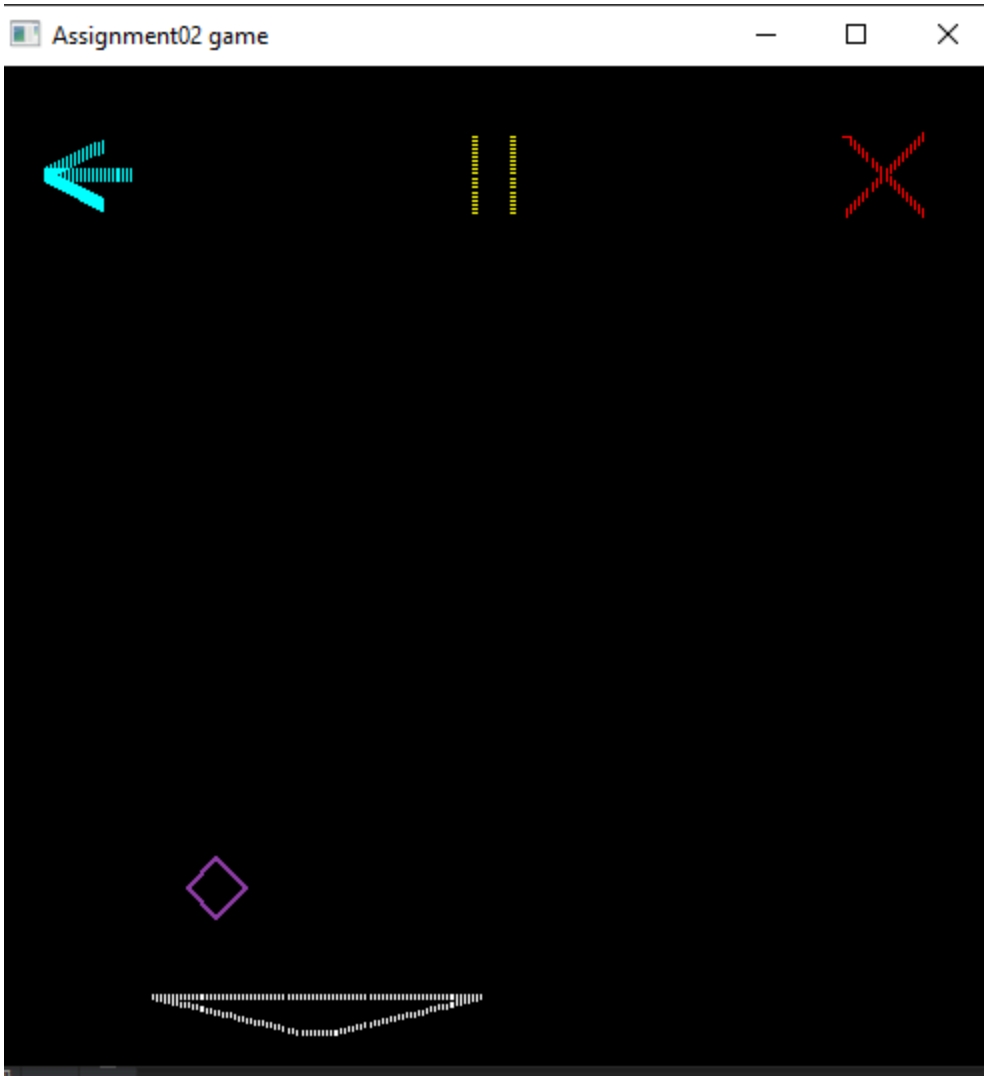

Fig: Catch the Diamonds Game

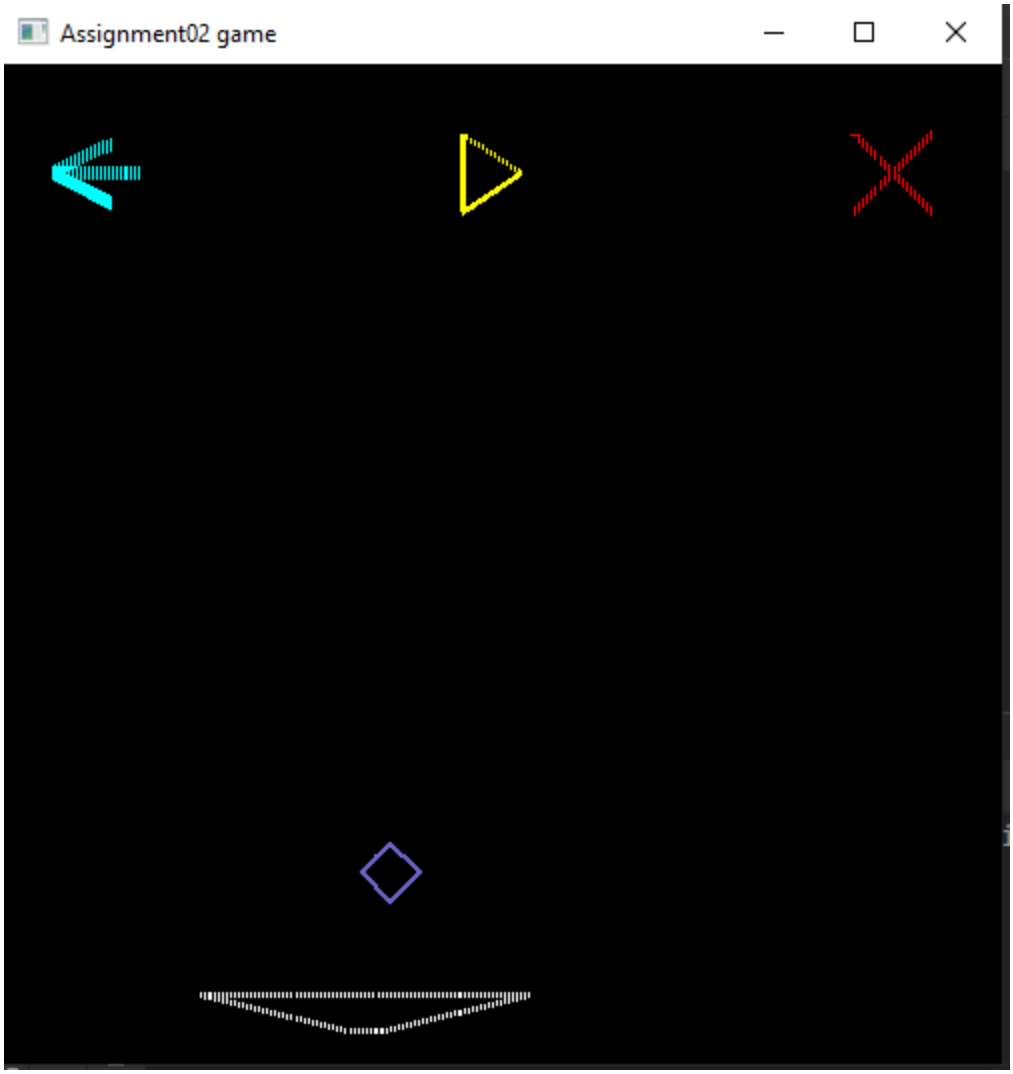

Fig: Game is paused

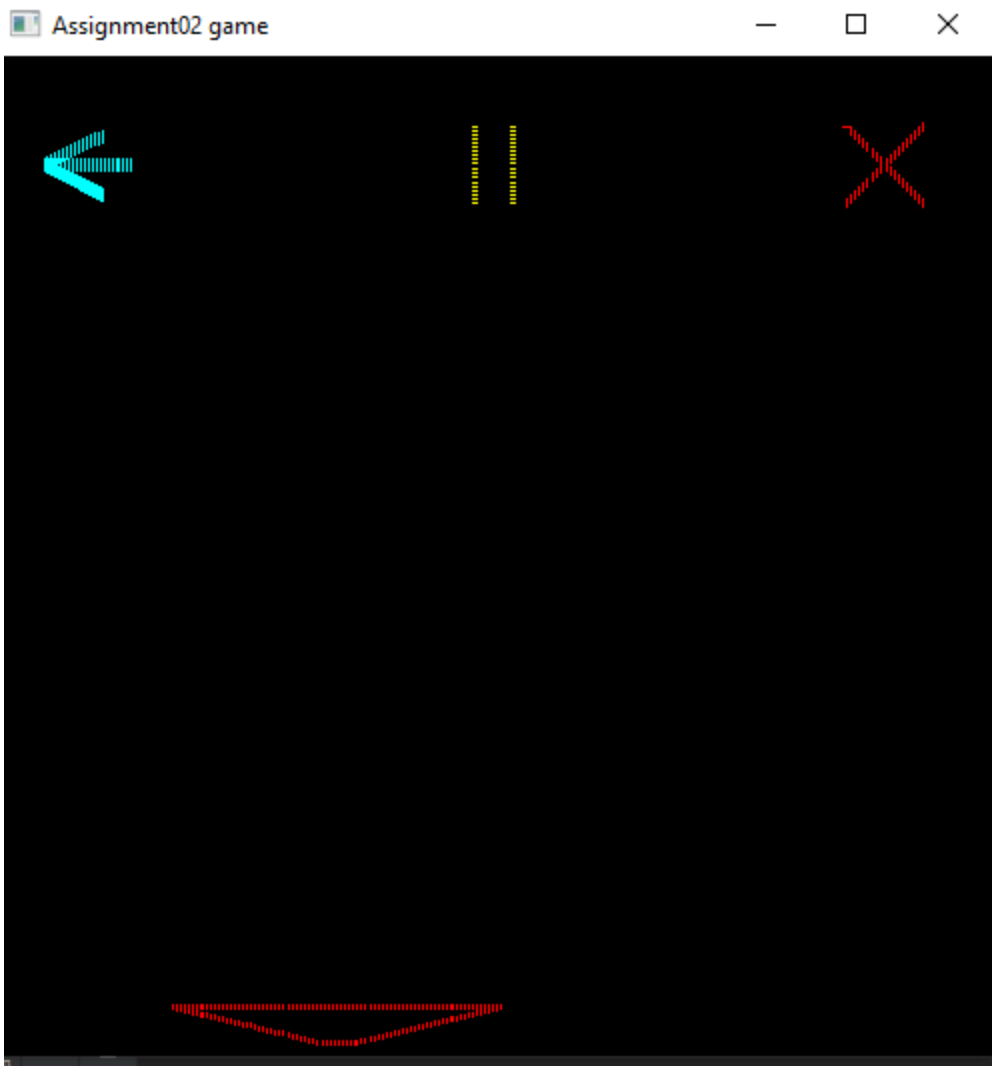

Fig: Game over (catcher turned red)