```python
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import random
import math
import sys


shooter_x = 0
shooter_y = -0.8
projectiles = []
falling_circles = []
score = 0
missed = 0
misfires = 0
is_game_over = False
paused = False

circle_radius = 0.05
max_misses = 3
window_width = 800
window_height = 600

# Midpoint Circle Drawing
def draw_circle(xc, yc, radius):
    x = 0
    y = radius
    d = 1 - radius
    glBegin(GL_POINTS)
    while x <= y:
        plot_circle_points(xc, yc, x, y)
        if d < 0:
            d = d + 2 * x + 3
        else:
            d = d + 2 * (x - y) + 5
            y -= 1
        x += 1
    glEnd()

def plot_circle_points(xc, yc, x, y):
```

```
    points = [
        (x, y), (-x, y), (x, -y), (-x, -y),
        (y, x), (-y, x), (y, -x), (-y, -x)
    ]
    for px, py in points:
        glVertex2f(xc + px / (window_width / 2), yc + py / (window_height
/ 2))

def spawn_circle():
    if len(falling_circles) < 5 and not is_game_over:
        x = random.uniform(-1, 1)
        falling_circles.append([x, 1.0])

def shoot_projectile():
    global misfires, is_game_over
    if not paused and not is_game_over:
        projectiles.append([shooter_x, shooter_y + circle_radius])
        misfires += 1
        if misfires >= max_misses:
            print("Game Over! Misfires: ", misfires)
            is_game_over = True

def update_falling_circles():
    global missed, is_game_over
    for circle in falling_circles:
        circle[1] -= 0.01
        if circle[1] < shooter_y:
            missed += 1
            falling_circles.remove(circle)
            if missed >= max_misses:
                print("Game Over! Missed Circles: ", missed)
                is_game_over = True

def update_projectiles():
    global score, misfires
    for projectile in projectiles:
        projectile[1] += 0.02
        if projectile[1] > 1.0:
            projectiles.remove(projectile)
```

```
        for circle in falling_circles:
            if (math.sqrt((projectile[0] - circle[0]) ** 2 +
(projectile[1] - circle[1]) ** 2) < circle_radius):
                score += 1
                projectiles.remove(projectile)
                falling_circles.remove(circle)
                misfires -= 1
                break


def special_input(key, x, y):
    global shooter_x
    if key == GLUT_KEY_LEFT and shooter_x > -1 + circle_radius:
        shooter_x -= 0.1
    if key == GLUT_KEY_RIGHT and shooter_x < 1 - circle_radius:
        shooter_x += 0.1
    glutPostRedisplay()

def keyboard(key, x, y):
    if key == b' ':
        shoot_projectile()

def draw_shooter():
    glColor3f(0.0, 0.0, 1.0)
    draw_circle(shooter_x, shooter_y, circle_radius)

def draw_projectiles():
    glColor3f(1.0, 0.0, 0.0)
    for projectile in projectiles:
        draw_circle(projectile[0], projectile[1], 0.02)

def draw_falling_circles():
    glColor3f(0.0, 1.0, 0.0)
    for circle in falling_circles:
        draw_circle(circle[0], circle[1], circle_radius)

def display():
    glClear(GL_COLOR_BUFFER_BIT)
    if not is_game_over:
        draw_shooter()
```

```
        draw_projectiles()
        draw_falling_circles()
    glutSwapBuffers()

def update(value):
    global is_game_over
    if not paused and not is_game_over:
        spawn_circle()
        update_falling_circles()
        update_projectiles()
    glutPostRedisplay()
    glutTimerFunc(16, update, 0)

def mouse_click(button, state, x, y):
    global paused, is_game_over
    if button == GLUT_LEFT_BUTTON and state == GLUT_DOWN:
        if 50 <= x <= 100 and 50 <= y <= 100:
            restart_game()
        elif 110 <= x <= 160 and 50 <= y <= 100:
            paused = not paused
        elif 170 <= x <= 220 and 50 <= y <= 100:
            print("Goodbye! Final Score: ", score)
            sys.exit(0)

def restart_game():
    global score, missed, misfires, is_game_over, projectiles,
falling_circles
    score = 0
    missed = 0
    misfires = 0
    is_game_over = False
    projectiles = []
    falling_circles = []
    print("Starting Over...")

def init():
    glClearColor(0.0, 0.0, 0.0, 1.0)
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0)

def main():
```

```
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
    glutInitWindowSize(window_width, window_height)
    glutCreateWindow(b"Mayeesha_21201462")

    init()
    glutDisplayFunc(display)
    glutSpecialFunc(special_input)
    glutKeyboardFunc(keyboard)
    glutMouseFunc(mouse_click)
    glutTimerFunc(16, update, 0)
    glutMainLoop()

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