*from* OpenGL.GL *import* \*

*from* OpenGL.GLUT *import* \*

*from* OpenGL.GLU *import* \*

*import* random

*# Constants*

SCREEN\_WIDTH = 800

SCREEN\_HEIGHT = 600

CATCHER\_WIDTH = 150

CATCHER\_HEIGHT = 30

FALL\_SPEED = 2

NUM\_DIAMONDS = 10

RETRY\_BUTTON\_LOCATION = (20, SCREEN\_HEIGHT - 50)

PAUSE\_BUTTON\_LOCATION = (SCREEN\_WIDTH/2, SCREEN\_HEIGHT - 50)

EXIT\_BUTTON\_LOCATION = (SCREEN\_WIDTH - 50, SCREEN\_HEIGHT - 50)

*# Colors*

WHITE = (1.0, 1.0, 1.0)

RED = (1.0, 0.0, 0.0)

TEAL = (0.0, 1.0, 1.0)

AMBER = (1.0, 0.5, 0.0)

GREEN = (0.0, 1.0, 0.0)

BLUE = (0.0, 0.0, 1.0)

PURPLE = (1.0, 0.0, 1.0)

YELLOW = (1.0, 1.0, 0.0)

colors = [RED, TEAL, AMBER, GREEN, BLUE, PURPLE, YELLOW]

*# initial state*

game\_over = False

paused = False

exit\_game = False

catcher\_color = WHITE

retry\_color = TEAL

pause\_color = AMBER

exit\_color = RED

catcher\_x = SCREEN\_WIDTH / 2

catcher\_y = 30

diamond\_x = random.randint(0, SCREEN\_WIDTH)

diamond\_y = SCREEN\_HEIGHT - 10

diamonds = []

falling\_diamond = None

current\_fall\_speed = FALL\_SPEED

score = 0

*# draw functions*

def draw\_retry\_button(x, y, color = retry\_color):

draw\_line(x, y, x + 20, y - 20, color)

draw\_line(x, y, x + 20, y + 20, color)

draw\_line(x, y, x + 50, y, color)

def draw\_pause\_button(x, y, color = pause\_color):

draw\_line(x + 10, y + 20, x + 10, y - 20, color)

draw\_line(x - 10, y + 20, x - 10, y - 20, color)

def draw\_play\_button(x, y, color = pause\_color):

draw\_line(x - 10, y + 20, x - 10, y - 20, color)

draw\_line(x - 10, y + 20, x + 10, y, color)

draw\_line(x - 10, y - 20, x + 10, y, color)

def draw\_exit(x, y, color = exit\_color):

draw\_line(x - 10, y + 10, x + 10, y - 10, color)

draw\_line(x - 10, y - 10, x + 10, y + 10, color)

def draw\_diamond(x, y, color):

width = 15/2

height = 30/2

draw\_line(x, y, x - width, y + height, color)

draw\_line(x, y, x + width, y + height, color)

draw\_line(x, y + (2\*height), x - width, y + height, color)

draw\_line(x, y + (2\*height), x + width, y + height, color)

def draw\_catcher():

x1, x2 = catcher\_x - CATCHER\_WIDTH / 2, catcher\_x + CATCHER\_WIDTH / 2

x3, x4 = x1 + 20, x2 - 20

y1, y2 = catcher\_y, catcher\_y - 20

draw\_line(x1, y1, x2, y1, catcher\_color)

draw\_line(x3, y2, x4, y2, catcher\_color)

draw\_line(x2, y1, x4, y2, catcher\_color)

draw\_line(x1, y1, x3, y2, catcher\_color)

*# animate functions*

def animate(v):

global falling\_diamond, score, catcher\_x, game\_over, catcher\_color, current\_fall\_speed, paused

*if* not game\_over and not paused:

*if* not falling\_diamond:

*# If there's no currently falling diamond and there are diamonds left*

*if* diamonds:

falling\_diamond = diamonds.pop(0)

*# If there's a currently falling diamond, update its position*

*if* falling\_diamond:

diamond\_x, diamond\_y, diamond\_color = falling\_diamond

diamond\_y -= current\_fall\_speed

falling\_diamond = (diamond\_x, diamond\_y, diamond\_color)

*if* collision(diamond\_y, diamond\_x, catcher\_x):

handle\_collision()

*# Game over when a diamond is missed*

*elif* diamond\_y < 0:

game\_over = True

falling\_diamond = None

catcher\_color = RED

current\_fall\_speed = FALL\_SPEED

print("Game Over! Score:",score)

*if* exit\_game:

glutLeaveMainLoop()

glutPostRedisplay()

glutTimerFunc(10, animate, 0)

*# main display function*

def display():

glClear(GL\_COLOR\_BUFFER\_BIT)

draw\_catcher()

*if* falling\_diamond:

x, y, color = falling\_diamond

draw\_diamond(x, y, color)

draw\_retry\_button(RETRY\_BUTTON\_LOCATION[0], RETRY\_BUTTON\_LOCATION[1])

draw\_exit(EXIT\_BUTTON\_LOCATION[0], EXIT\_BUTTON\_LOCATION[1])

*if* not paused: draw\_pause\_button(PAUSE\_BUTTON\_LOCATION[0], PAUSE\_BUTTON\_LOCATION[1])

*else*: draw\_play\_button(PAUSE\_BUTTON\_LOCATION[0], PAUSE\_BUTTON\_LOCATION[1])

glutSwapBuffers()

*# input handler*

def handle\_mouse(button, state, x, y):

global paused, exit\_game

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

*# handle restart button click*

*if* x >= RETRY\_BUTTON\_LOCATION[0] and x <= RETRY\_BUTTON\_LOCATION[0] + 50:

y = SCREEN\_HEIGHT - y

*if* y >= (RETRY\_BUTTON\_LOCATION[1] - 20) and y <= (RETRY\_BUTTON\_LOCATION[1] + 20):

print("Retry")

handle\_retry()

*# handle pause button click*

*elif* x >= PAUSE\_BUTTON\_LOCATION[0] - 10 and x <= PAUSE\_BUTTON\_LOCATION[0] + 10:

y = SCREEN\_HEIGHT - y

*if* y >= (PAUSE\_BUTTON\_LOCATION[1] - 20) and y <= (PAUSE\_BUTTON\_LOCATION[1] + 20):

*if* not paused: print("Paused")

*else*: print("Play")

paused = not paused

*# handle EXIT button click*

*elif* x >= EXIT\_BUTTON\_LOCATION[0] - 10 and x <= EXIT\_BUTTON\_LOCATION[0] + 10:

y = SCREEN\_HEIGHT - y

*if* y >= (EXIT\_BUTTON\_LOCATION[1] - 10) and y <= (EXIT\_BUTTON\_LOCATION[1] + 10):

exit\_game = True

def handle\_keyboard(key, x, y):

global catcher\_x

speed = 20

*if* key == GLUT\_KEY\_LEFT and not game\_over and not paused:

catcher\_x -= speed

*if* catcher\_x < CATCHER\_WIDTH / 2:

catcher\_x = CATCHER\_WIDTH / 2

*elif* key == GLUT\_KEY\_RIGHT and not game\_over and not paused:

catcher\_x += speed

*if* catcher\_x > SCREEN\_WIDTH - CATCHER\_WIDTH / 2:

catcher\_x = SCREEN\_WIDTH - CATCHER\_WIDTH / 2

*# event handler*

def handle\_retry ():

global score, lives, game\_over, catcher\_color, falling\_diamond, paused

paused = False

score = 0

lives = 1

game\_over = False

catcher\_color = WHITE

diamonds.clear()

falling\_diamond = None

*for* \_ *in* range(NUM\_DIAMONDS):

generate\_diamond()

def collision(diamond\_y, diamond\_x, catcher\_x):

*return* diamond\_y <= CATCHER\_HEIGHT and abs(diamond\_x - catcher\_x) < CATCHER\_WIDTH / 2

def handle\_collision():

global score, falling\_diamond, current\_fall\_speed

score += 1

print("Score:",score)

falling\_diamond = None

current\_fall\_speed += 0.5

*# diamond generator*

def generate\_diamond():

*if* not game\_over:

new\_diamond\_x = random.randint(15, SCREEN\_WIDTH - 15)

new\_diamond\_y = SCREEN\_HEIGHT - 15

color = random.choice(colors) *# Choose a random color*

diamonds.append((new\_diamond\_x, new\_diamond\_y, color))

*# MidPoint Line Drawing Algorithm*

def draw\_points(x, y, color = (1, 1, 1), size=2):

glColor3fv(color)

glPointSize(size)

glBegin(GL\_POINTS)

glVertex2f(x,y)

glEnd()

def to\_zone0(zone, x, y):

*if* zone == 0: *return* (x,y)

*elif* zone == 1: *return* (y,x)

*elif* zone == 2: *return* (y,-x)

*elif* zone == 3: *return* (-x,y)

*elif* zone == 4: *return* (-x,-y)

*elif* zone == 5: *return* (-y,-x)

*elif* zone == 6: *return* (-y,x)

*elif* zone == 7: *return* (x,-y)

*else*: *raise* ValueError("Zone must be in [0, 7]")

def to\_zoneM(zone, x, y):

*if* zone == 0: *return* (x,y)

*elif* zone == 1: *return* (y,x)

*elif* zone == 2: *return* (-y,x)

*elif* zone == 3: *return* (-x,y)

*elif* zone == 4: *return* (-x,-y)

*elif* zone == 5: *return* (-y,-x)

*elif* zone == 6: *return* (y,-x)

*elif* zone == 7: *return* (x,-y)

*else*: *raise* ValueError("Zone must be in [0, 7]")

def find\_zone(x1,y1,x2,y2):

dx = x2 - x1

dy = y2 - y1

*if* abs(dx) > abs(dy):

*if* dx>=0 and dy>=0: *return* 0

*elif* dx>=0 and dy<=0: *return* 7

*elif* dx<=0 and dy>=0: *return* 3

*elif* dx<=0 and dy<=0: *return* 4

*else* :

*if* dx>=0 and dy>=0: *return* 1

*elif* dx<=0 and dy>=0: *return* 2

*elif* dx<=0 and dy<=0: *return* 5

*elif* dx>=0 and dy<=0: *return* 6

def draw\_line(x1, y1, x2, y2, color):

zone = find\_zone(x1,y1,x2,y2)

x1,y1 = to\_zone0(zone, x1, y1)

x2,y2 = to\_zone0(zone, x2, y2)

dx = x2 - x1

dy = y2 - y1

d = 2\*dy - dx

incrE = 2\*dy

incrNE = 2\*(dy - dx)

x = x1

y = y1

x0, y0 = to\_zoneM(zone, x, y)

draw\_points(x0, y0, color)

*while* x < x2:

*if* d <= 0:

d = d + incrE

x = x + 1

*else*:

d = d + incrNE

x = x + 1

y = y + 1

x0, y0 = to\_zoneM(zone, x, y)

draw\_points(x0, y0, color)

*# main function*

def main():

glutInit()

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)

glutInitWindowSize(SCREEN\_WIDTH, SCREEN\_HEIGHT)

glutCreateWindow(b"Catch the Diamonds")

*# Initialize OpenGL*

glClearColor(0.0, 0.0, 0.0, 1.0)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

glOrtho(0, SCREEN\_WIDTH, 0, SCREEN\_HEIGHT, -1, 1)

glMatrixMode(GL\_MODELVIEW)

*# Register callback functions*

glutDisplayFunc(display)

glutTimerFunc(10, animate, 0)

glutSpecialFunc(handle\_keyboard)

glutMouseFunc(handle\_mouse)

*for* \_ *in* range(NUM\_DIAMONDS):

generate\_diamond()

glutMainLoop()

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

main()





