**Catch the Diamonds**

**Codes:**

Save the codes according to their file name as shown above each code. Then just run the `game.py` file to play the game.

**LineModule.py**

#!/usr/bin/env python3

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

class Line:

def \_\_init\_\_(self, x0, y0, x1, y1, size = 1, color = (1, 1, 1)):

self.size = size

self.color = color

self.zoneZero = {

0: lambda x, y: (x, y),

1: lambda x, y: (y, x),

2: lambda x, y: (y, -x),

3: lambda x, y: (-x, y),

4: lambda x, y: (-x, -y),

5: lambda x, y: (-y, -x),

6: lambda x, y: (-y, x),

7: lambda x, y: (x, -y),

}

self.zoneOriginal = {

0: lambda x, y: (x, y),

1: lambda x, y: (y, x),

2: lambda x, y: (-y, x),

3: lambda x, y: (-x, y),

4: lambda x, y: (-x, -y),

5: lambda x, y: (-y, -x),

6: lambda x, y: (y, -x),

7: lambda x, y: (x, -y),

}

self.zone = self.findZone(x0, y0, x1, y1)

# print(f"{self.zone} -> zone")

x0, y0 = self.zoneZero[self.zone](x0, y0)

x1, y1 = self.zoneZero[self.zone](x1, y1)

# print(f"({x0}, {y0}) ({x1}, {y1}) -> converted")

self.midPointAlgo(x0, y0, x1, y1)

def findZone(self, x0, y0, x1, y1):

dy = y1 - y0

dx = x1 - x0

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

if (dx >= 0 and dy >= 0):

return 0

elif (dx <= 0 and dy >= 0):

return 3

elif (dx <= 0 and dy <= 0):

return 4

else:

return 7

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

else:

return 6

def drawPoint(self, x, y):

glPointSize(self.size)

glBegin(GL\_POINTS)

glColor3f(self.color[0], self.color[1], self.color[2])

glVertex2f(x, y)

glEnd()

def midPointAlgo(self, x0, y0, x1, y1):

dy = y1 - y0

dx = x1 - x0

d\_init = 2\*dy - dx

incE = 2\*dy

incNE = 2\*(dy - dx)

while (x0 <= x1):

a, b = self.zoneOriginal[self.zone](x0, y0)

self.drawPoint(a, b)

# print(f"{a}, {b} done")

x0 += 1

if d\_init <= 0:

d\_init += incE

else:

y0 += 1

d\_init += incNE

**Components.py**

#!/usr/bin/env python3

from LineModule import Line

import random

class Diamond:

def \_\_init\_\_(self, x, y, d=25, width=3):

self.x = x

self.y = y

self.d = d

self.color = (random.random(), random.random(), random.random())

self.width = 3

def draw(self):

Line(self.x, self.y + self.d, self.x + self.d, self.y, self.width, self.color)

Line(self.x, self.y + self.d, self.x - self.d, self.y, self.width, self.color)

Line(self.x - self.d, self.y, self.x, self.y - self.d, self.width, self.color)

Line(self.x + self.d, self.y, self.x, self.y - self.d, self.width, self.color)

def getRightVertex(self):

vr\_x = self.x + self.d

vr\_y = self.y - self.d

return vr\_x, vr\_y

def getLeftVertex(self):

vl\_x = self.x - self.d

vl\_y = self.y - self.d

return vl\_x, vl\_y

class Pot:

def \_\_init\_\_(self, x=250, y=20, width=2):

self.x = x

self.y = y

self.height = 30

self.width = width

def draw(self, color=(1, 1, 1)):

Line(self.x+50, self.y, self.x-50, self.y, self.width, color)

Line(self.x+50, self.y, self.x+70, self.y+30, self.width, color)

Line(self.x-50, self.y, self.x-70, self.y+30, self.width, color)

Line(self.x+70, self.y+30, self.x-70, self.y+30, self.width, color)

def getRightVertex(self):

vr\_x = self.x + 70

vr\_y = self.y + self.height

return vr\_x, vr\_y

def getLeftVertex(self):

vl\_x = self.x - 70

vl\_y = self.y + self.height

return vl\_x, vl\_y

class Arrow:

def \_\_init\_\_(self, x=70, y=760, d=25, width=3, color=(0, 0.705, 0.705)):

self.x = x

self.y = y

self.d = d

self.width = width

self.color = color

self.vertex\_x = self.x - self.d

self.vertex\_y = self.y

def draw(self):

Line(self.x, self.y, self.x + self.d, self.y, self.width, self.color)

Line(self.x, self.y, self.vertex\_x, self.vertex\_y, self.width, self.color)

Line(self.vertex\_x, self.vertex\_y, self.x, self.y + self.d, self.width, self.color)

Line(self.vertex\_x, self.vertex\_y, self.x, self.y - self.d, self.width, self.color)

class Pause:

def \_\_init\_\_(self, x=240, y=760, d=25, width=3, color=(0.125, 0.660, 0.286)):

self.x = x

self.y = y

self.d = d

self.width = width

self.space = 15

self.color = color

def draw(self):

Line(self.x + self.space, self.y + self.d, self.x + self.space, self.y - self.d, self.width, self.color)

Line(self.x - self.space, self.y + self.d, self.x - self.space, self.y - self.d, self.width, self.color)

class Play:

def \_\_init\_\_(self, x=240, y=760, d=25, width=3, color=(0.125, 0.660, 0.286)):

self.x = x

self.y = y

self.d = d

self.width = width

self.color = color

self.vertex\_x = self.x + self.d

self.vertex\_y = self.y

def draw(self):

Line(self.x, self.y + self.d, self.x, self.y - self.d, self.width, self.color)

Line(self.vertex\_x, self.y, self.x, self.y + self.d, self.width, self.color)

Line(self.vertex\_x, self.y, self.x, self.y - self.d, self.width, self.color)

class Cross:

def \_\_init\_\_(self, x=400, y=760, d=25, width=3, color=(0.840, 0.0924, 0.0924)):

self.x = x

self.y = y

self.d = d

self.width = width

self.color = color

def draw(self):

Line(self.x, self.y, self.x + self.d, self.y + self.d, self.width, self.color)

Line(self.x, self.y, self.x + self.d, self.y - self.d, self.width, self.color)

Line(self.x, self.y, self.x - self.d, self.y + self.d, self.width, self.color)

Line(self.x, self.y, self.x - self.d, self.y - self.d, self.width, self.color)

**game.py**

#!/usr/bin/env python3

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

######## CSE423\_Lab02 ####################

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

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

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

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

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from LineModule import Line

from Components import \*

import random

import time

class Game:

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

self.width = 500

self.height = 800

self.pot\_speed = 100

self.diamond\_speed = 8

self.pot = Pot()

self.diamond = None

self.score = 0

self.freeze = False

self.gameOver = False

self.prev\_time = time.time()

self.dt = 0

self.fps = 30

def calcDeltaTime(self):

self.dt = time.time() - self.prev\_time

self.prev\_time = time.time()

self.dt \*= self.fps

def generateDiamond(self):

if not self.diamond:

self.diamond = Diamond(x=random.randint(25, 475), y=710)

def checkBoundary(self):

if self.diamond and self.diamond.y <= 0:

self.diamond = None

self.gameOver = True

print(f"Game Over! Score: {self.score}")

def collisionCheck(self):

if self.diamond and self.pot:

d\_vl\_x, d\_vl\_y = self.diamond.getLeftVertex()

d\_vr\_x, d\_vr\_y = self.diamond.getRightVertex()

p\_vl\_x, p\_vl\_y = self.pot.getLeftVertex()

p\_vr\_x, p\_vr\_y = self.pot.getRightVertex()

if p\_vl\_y >= d\_vr\_y or p\_vr\_y >= d\_vl\_y:

if (d\_vl\_x <= p\_vr\_x and d\_vl\_x >= p\_vl\_x) or (d\_vr\_x >= p\_vl\_x and d\_vr\_x <= p\_vr\_x):

# self.diamond\_speed += 0.5

self.diamond = None

self.score += 1

self.diamond\_speed += 0.25

print(f"Score: {self.score}")

def drawComp(self):

# Line(0, 720, 500, 720)

# Line(160, 720, 160, 800)

# Line(320, 720, 320, 800)

if self.diamond:

self.diamond.draw()

Arrow().draw()

Pause().draw() if not self.freeze else Play().draw()

Cross().draw()

self.pot.draw((0.941, 0.086, 0.086)) if self.gameOver else self.pot.draw()

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(0, 0, 0, 1.0)

self.calcDeltaTime()

if not self.gameOver:

self.generateDiamond()

self.drawComp()

self.collisionCheck()

self.checkBoundary()

glutSwapBuffers()

def animate(self):

if self.diamond and not self.freeze:

self.diamond.y -= self.diamond\_speed \* self.dt

glutPostRedisplay()

def checkButton(self, x, y):

if (y >= 720 and y <= 800):

if (x <= 160 and x >= 0):

self.score = 0

self.gameOver = False

self.diamond = None

print("Starting Over!")

elif (x <= 320 and x >= 165):

self.freeze = True if not self.freeze else False

elif (x <= 500 and x >= 325):

glutLeaveMainLoop()

else:

print("Can't Recognize")

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

pass

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

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

self.checkButton(x, self.height - y)

glutPostRedisplay()

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

if not self.gameOver and not self.freeze:

if key == GLUT\_KEY\_LEFT:

if not self.pot.getLeftVertex()[0] <= 0:

self.pot.x -= self.pot\_speed \* self.dt

if key == GLUT\_KEY\_RIGHT:

if not self.pot.getRightVertex()[0] >= self.width:

self.pot.x += self.pot\_speed \* self.dt

glutPostRedisplay()

def run(self):

glutInit()

glutInitDisplayMode(GLUT\_RGBA)

glutInitWindowSize(self.width, self.height)

glutInitWindowPosition(0, 0)

glutCreateWindow("OpenGL Catch The Diamond")

glutDisplayFunc(self.showScreen)

glutIdleFunc(self.animate)

glutKeyboardFunc(self.keyboardListener)

glutMouseFunc(self.mouseListener)

glutSpecialFunc(self.specialKeyListener)

glutMainLoop()

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

Game().run()

**Functionalities:**

Note that all the functionalities have been shown in the single gif file. Please locate it below.

