```
import tkinter as tk
import turtle as trtl
import random
import math
import time

# sidenote: i understand i don't use the usual python naming conventions;
this was admittedly intentional.
# in any case, this is pong with a bit of flair

# initializing basic variables
wn = trtl.Screen()
paddleVelocity = 14
ballVelocity = 18
angleGoing = 0
lastKeyPress = ""
inputCounter = 0
enemyPaddleFreezeRadInitial = 15
enemyPaddleFreezeRad = enemyPaddleFreezeRadInitial
userScore = 0
enemyScore = 0
# color list! i needed a list. :P
colorList = ["red", "blue", "white", "orange", "yellow"]
enemyColor = 0
userColor = 1
ballColor = 2

# decorative shapes
arena = trtl.Turtle()
arena.shape("square")
arena.turtlesize(21)
arena.speed(0)
arena.back(20)

arenaTwo = trtl.Turtle()
arenaTwo.shape("square")
arenaTwo.turtlesize(21)
arenaTwo.speed(0)
arenaTwo.forward(20)
```

```
centerline = trtl.Turtle()
centerline.speed(0)
centerline.color("white")
centerline.fillcolor("white")
centerline.shape("square")
centerline.shapesize(stretch_wid=20, stretch_len=0.06)

middle = trtl.Turtle()
middle.turtlesize(0.5)
middle.shape("circle")
middle.color("white")
middle.fillcolor("white")

# create left paddle
enemyPaddle = trtl.Turtle()
enemyPaddle.shape("square")
enemyPaddle.fillcolor("red")
enemyPaddle.color("red")
enemyPaddle.penup()
enemyPaddle.speed(0)
enemyPaddle.goto(-220, 0)
enemyPaddle.shapesize(stretch_wid = 4, stretch_len = 1)

# create right paddle
userPaddle = trtl.Turtle()
userPaddle.shape("square")
userPaddle.fillcolor("blue")
userPaddle.color("blue")
userPaddle.penup()
userPaddle.speed(0)
userPaddle.goto(220, 0)
userPaddle.shapesize(stretch_wid = 4, stretch_len = 1)

# create scoring for user
userScorer = trtl.Turtle()
userScorer.turtlesize(0.01)
userScorer.pencolor("blue")
userScorer.penup()
userScorer.goto(60, 150)
userScorer.pendown()
```

```
userScorer.speed(0)
userScorer.write(str(0), align="center", font=("Terminal", 32, "bold"))

# create scoring for enemy
enemyScorer = trtl.Turtle()
enemyScorer.turtlesize(0.01)
enemyScorer.pencolor("red")
enemyScorer.penup()
enemyScorer.goto(-60, 150)
enemyScorer.pendown()
enemyScorer.speed(0)
enemyScorer.write(str(0), align="center", font=("Terminal", 32, "bold"))

# create ball
pongBall = trtl.Turtle()
pongBall.shape("circle")
pongBall.color("white")
pongBall.fillcolor("white")
pongBall.speed(0)
pongBall.penup()

# create direction indicator
dirIndicator = trtl.Turtle()
dirIndicator.fillcolor("red")
dirIndicator.color("red")
dirIndicator.pencolor("red")
dirIndicator.pensize(6)
dirIndicator.turtlesize(0.01)
dirIndicator.speed(0)

# initialize paddle coordinates
enemyPaddleX = enemyPaddle.xcor()
enemyPaddleY = enemyPaddle.ycor()

userPaddleX = userPaddle.xcor()
userPaddleY = userPaddle.ycor()

randomStart = random.randint(1,4)
```

```
# initialize ball direction and show it with arrow; same as RandomAngle
function
if (randomStart == 1):
    angleGoing = random.randint(40, 60)
elif (randomStart == 2):
    angleGoing = random.randint(120, 140)
elif (randomStart == 3):
    angleGoing = random.randint(220, 240)
else:
    angleGoing = random.randint(300, 320)
dirIndicator.seth(angleGoing)
dirIndicator.penup()
dirIndicator.forward(13)
dirIndicator.pendown()
dirIndicator.forward(40)
dirIndicator.turtlesize(2)
dirIndicator.penup()
dirIndicator.forward(12)
angleGoing = float(math.radians(angleGoing))
xVelocity = float(math.cos(angleGoing) * ballVelocity)
yVelocity = float(math.sin(angleGoing) * ballVelocity)

# used to set random angle for the ball and show it with arrow
def RandomAngle(min, max):
    global angleGoing
    global xVelocity
    global yVelocity
    global dirIndicator
    global pongBall
    # sets a random angle for the ball, then indicates it with an arrow
    angleGoing = random.randint(min, max)
    dirIndicator.goto(pongBall.xcor(), pongBall.ycor())
    dirIndicator.seth(angleGoing)
    dirIndicator.forward(13)
    dirIndicator.pendown()
    dirIndicator.forward(40)
    dirIndicator.turtlesize(2)
    dirIndicator.penup()
    dirIndicator.forward(12)
    # converts angle to radians and sets velocities accordingly
```

```
    angleGoing = float(math.radians(angleGoing))
    xVelocity = float(math.cos(angleGoing) * ballVelocity)
    yVelocity = float(math.sin(angleGoing) * ballVelocity)

# moves user paddle up
def MoveUp():
    global userPaddle
    global lastKeyPress
    global inputCounter
    global userPaddleX
    global userPaddleY
    # vv buffer to ensure the user can't just move before the game even
updates
    if (inputCounter <= 3):
        if (userPaddleY < 160):
            userPaddle.goto(userPaddle.xcor(), userPaddle.ycor() +
paddleVelocity * 1.25)
        # records the last input
        lastKeyPress = "Up"
        # 4 frames given to either sharpen or flatten angle depending on
input
        inputCounter = 4

    userPaddleX = userPaddle.xcor()
    userPaddleY = userPaddle.ycor()

# moves user paddle down
def MoveDown():
    global userPaddle
    global lastKeyPress
    global inputCounter
    global userPaddleX
    global userPaddleY
    # vv buffer to ensure the user can't just move before the game even
updates
    if (inputCounter <= 3):
        if (userPaddleY > -160):
            userPaddle.goto(userPaddle.xcor(), userPaddle.ycor() -
paddleVelocity * 1.25)
        # records the last input
```

```
        lastKeyPress = "Down"
        # 4 frames given to either sharpen or flatten angle depending on
input
        inputCounter = 4

    userPaddleX = userPaddle.xcor()
    userPaddleY = userPaddle.ycor()

# flattens the angle of the ball
def Flatten():
    global xVelocity
    global yVelocity
    # reduces y velocity and increases x velocity to flatten angle
    xVelocity = xVelocity * 1.5
    yVelocity = yVelocity / 1.3

# steepens the angle of the ball
def Steepen():
    global xVelocity
    global yVelocity
    # reduces x velocity and increases y velocity to sharpen angle
    xVelocity = xVelocity / 1.3
    yVelocity = yVelocity * 1.5

# updates the score when the game pauses after the ball is scored
def UpdateScores():
    global enemyScore
    global enemyScorer
    global userScore
    global userScorer
    # writes down the new scores for the game
    enemyScorer.clear()
    enemyScorer.write(str(enemyScore), align="center", font=("Terminal",
32, "bold"))
    userScorer.clear()
    userScorer.write(str(userScore), align="center", font=("Terminal", 32,
"bold"))

def CycleEnemyColor():
    global enemyPaddle
```

```
    global enemyColor
    global enemyScorer
    global colorList
    # cycles enemy color
    enemyColor = enemyColor + 1
    enemyColor = enemyColor % len(colorList)
    enemyPaddle.fillcolor(colorList[enemyColor])
    enemyPaddle.color(colorList[enemyColor])
    enemyScorer.pencolor(colorList[enemyColor])

def CycleUserColor():
    global userPaddle
    global userColor
    global colorList
    # cycles user color
    userColor = userColor + 1
    userColor = userColor % len(colorList)
    userPaddle.fillcolor(colorList[userColor])
    userPaddle.color(colorList[userColor])
    userScorer.pencolor(colorList[userColor])

def CycleBallColor():
    global pongBall
    global dirIndicator
    global ballColor
    global colorList
    # cycles ball color
    ballColor = ballColor + 1
    ballColor = ballColor % len(colorList)
    pongBall.fillcolor(colorList[ballColor])
    pongBall.color(colorList[ballColor])
    # cycles arrow color depending on color of ball
    ballColor = ballColor + 5
    dirIndicator.pencolor(colorList[(ballColor - 2) % 5])
    dirIndicator.color(colorList[(ballColor - 2) % 5])
    dirIndicator.fillcolor(colorList[(ballColor - 2) % 5])
    ballColor = ballColor - 5
```

```
# inputs, using up and down arrow keys to move the paddle up or down,
using ZXC to cycle colors
wn.onkeypress(MoveUp, "Up")
wn.onkeypress(MoveDown, "Down")
wn.onkeypress(CycleEnemyColor, "z")
wn.onkeypress(CycleUserColor, "x")
wn.onkeypress(CycleBallColor, "c")
wn.listen()
# pause before game begins
time.sleep(1.5)

while True:
    wn.update()
    # 25 FPS (allegedly.)
    time.sleep(float(0.04))

    # keep updating positions for both paddles
    enemyPaddleX = enemyPaddle.xcor()
    enemyPaddleY = enemyPaddle.ycor()

    # extra measure to make sure paddles do not go beyond bounds
    if (enemyPaddleY > 160):
        enemyPaddle.sety(160)
        enemyPaddleY = 160
    elif (enemyPaddleY < -160):
        enemyPaddle.sety(-160)
        enemyPaddleY = -160
    if (userPaddleY > 160):
        userPaddle.sety(160)
        userPaddleY = 160
    elif (userPaddleY < -160):
        userPaddle.sety(-160)
        userPaddleY = -160

    # counts down frames from the user's last input
    inputCounter -= 1

    # makes the paddle less likely to pause moving when the ball is
traveling at a sharp angle
    if (yVelocity > ballVelocity * 0.85):
```

```
        enemyPaddleFreezeRad = (enemyPaddleFreezeRadInitial *
enemyPaddleFreezeRadInitial) / yVelocity

    # ensures the turtle for the direction indicator isn't visible just in
case
    dirIndicator.turtlesize(0.01)
    dirIndicator.penup()
    dirIndicator.goto(0, 500)

    # automatically move enemy paddle towards ball's y-coordinate
    if (math.fabs(enemyPaddleY - pongBall.ycor()) > enemyPaddleFreezeRad):
        if (enemyPaddleY < pongBall.ycor() and enemyPaddleY < 160):
            if ((yVelocity > 0 and pongBall.ycor() < 165) or
(pongBall.xcor() > 155) or (pongBall.xcor() < -155)):
                enemyPaddle.goto(enemyPaddle.xcor(), enemyPaddle.ycor() +
paddleVelocity)
            else:
                # dampen paddle movement if the ball is moving opposite
the paddle's movement
                enemyPaddle.goto(enemyPaddle.xcor(), enemyPaddle.ycor() +
paddleVelocity * 0.7)
        elif (enemyPaddleY > pongBall.ycor() and enemyPaddleY > -160):
            if ((yVelocity < 0 and pongBall.ycor() > -165) or
(pongBall.xcor() > 155) or (pongBall.xcor() < -155)):
                enemyPaddle.goto(enemyPaddle.xcor(), enemyPaddle.ycor() -
paddleVelocity)
            else:
                enemyPaddle.goto(enemyPaddle.xcor(), enemyPaddle.ycor() -
paddleVelocity * 0.7)
    # move ball
    pongBall.goto(pongBall.xcor() + xVelocity, pongBall.ycor() +
yVelocity)
    # bounce ball off of floor and ceiling
    if (math.fabs(pongBall.ycor()) >= 200 and pongBall.ycor()/yVelocity >
0):
        yVelocity = yVelocity * -1
        pongBall.goto(pongBall.xcor(), 200 *
-(yVelocity/(math.fabs(yVelocity))))
        # check if x velocity is too small for the ball to go horizontally
quickly, then reset velocity with new angle and briefly pause game
```

```
        if (math.fabs(xVelocity) <= 0.16 * math.fabs(yVelocity)):
            if (yVelocity < 0):
                if (random.randint(1,2) == 1):
                    RandomAngle(210, 240)
                else:
                    RandomAngle(300, 330)
            elif (yVelocity > 0):
                if (random.randint(1,2) == 1):
                    RandomAngle(30, 60)
                else:
                    RandomAngle(120, 150)
            time.sleep(0.75)
    # check if the paddles are in range of the ball, if they are, bounce
the ball back, otherwise, reset positions and update scores before pausing
    if (math.fabs(pongBall.xcor()) >= 200):
        if (pongBall.xcor() < 0 and math.fabs(enemyPaddleY -
pongBall.ycor()) <= 55):
            xVelocity = xVelocity * -1
            # sets ball's position to in front of the paddles, one more
measure to make sure they stay going the opposite direction once hit
            pongBall.setx(-200)
            if (xVelocity < 0):
                xVelocity = xVelocity * -1
        elif (pongBall.xcor() > 0 and math.fabs(userPaddleY -
pongBall.ycor()) <= 55):
            pongBall.setx(200)
            if (xVelocity > 0):
                xVelocity = xVelocity * -1
            # flattens or steepens the ball if the user's last input was
close enough to when the ball was bounced back
            if (inputCounter > 0):
                # makes the ball's angle flatter if the user's last input
is opposite the direction of the ball's movement
                if ((lastKeyPress == "Down" and yVelocity > 0) or
(lastKeyPress == "Up" and yVelocity < 0)):
                    Flatten()
                # makes the ball's angle steeper if the user's last input
is in the same direction as the ball's movement
                elif ((lastKeyPress == "Up" and yVelocity > 0) or
(lastKeyPress == "Down" and yVelocity < 0)):
```

```
                Steepen()
        elif (math.fabs(pongBall.xcor()) >= 220):
            # reset paddle and ball positions
            pongBall.goto(0, 0)
            enemyPaddle.sety(0)
            enemyPaddleY = 0
            userPaddle.sety(0)
            userPaddleY = 0
            if (xVelocity > 0):
                # if the enemy scored, make the ball go to the player
                enemyScore += 1
                if (random.randint(1, 2) == 1):
                    RandomAngle (30, 60)
                else:
                    RandomAngle (300, 330)
            else:
                # if the player scored, make the ball go to the enemy
                userScore += 1
                if (random.randint(1,2) == 1):
                    RandomAngle (120, 150)
                else:
                    RandomAngle (210, 240)
            # update scores and pause game briefly
            UpdateScores()
            time.sleep(2)
```