#server

import socket

from \_thread import \*

import pickle

from game import Game

server = "127.0.0.1"

port = 7000

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

s.bind((server, port))

except socket.error as e:

str(e)

s.listen(2)

print("Waiting for a connection, Server Started")

connected = set()

games = {}

idCount = 0

def threaded\_client(conn, p, gameId):

global idCount

conn.send(str.encode(str(p)))

while True:

try:

data = conn.recv(4096).decode()

if gameId in games:

game = games[gameId]

if not data:

break

else:

if data == "reset":

game.resetWent()

elif data != "get":

game.play(p, data)

conn.sendall(pickle.dumps(game))

else:

break

except:

break

print("Lost connection")

try:

del games[gameId]

print("Closing Game", gameId)

except:

pass

idCount -= 1

conn.close()

while True:

conn, addr = s.accept()

print("Connected to:", addr)

idCount += 1

p = 0

gameId = (idCount - 1)//2

if idCount % 2 == 1:

games[gameId] = Game(gameId)

print("Creating a new game...")

else:

games[gameId].ready = True

p = 1

start\_new\_thread(threaded\_client, (conn, p, gameId))

#game

class Game:

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

self.p1Went = False

self.p2Went = False

self.ready = False

self.id = id

self.moves = [None, None]

self.wins = [0,0]

self.ties = 0

def get\_player\_move(self, p):

"""

:param p: [0,1]

:return: Move

"""

return self.moves[p]

def play(self, player, move):

self.moves[player] = move

if player == 0:

self.p1Went = True

else:

self.p2Went = True

def connected(self):

return self.ready

def bothWent(self):

return self.p1Went and self.p2Went

def winner(self):

p1 = self.moves[0].upper()[0]

p2 = self.moves[1].upper()[0]

winner = -1

if p1 == "R" and p2 == "S":

winner = 0

elif p1 == "S" and p2 == "R":

winner = 1

elif p1 == "P" and p2 == "R":

winner = 0

elif p1 == "R" and p2 == "P":

winner = 1

elif p1 == "S" and p2 == "P":

winner = 0

elif p1 == "P" and p2 == "S":

winner = 1

return winner

def resetWent(self):

self.p1Went = False

self.p2Went = False

#network

import socket

import pickle

class Network:

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

self.client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server = "127.0.0.1"

self.port = 7000

self.addr = (self.server, self.port)

self.p = self.connect()

def getP(self):

return self.p

def connect(self):

try:

self.client.connect(self.addr)

return self.client.recv(2048).decode()

except:

pass

def send(self, data):

try:

self.client.send(str.encode(data))

return pickle.loads(self.client.recv(2048\*2))

except socket.error as e:

print(e)

#client

import pygame

from network import Network

pygame.font.init()

width = 650

height = 650

win = pygame.display.set\_mode((width, height))

pygame.display.set\_caption("Client")

class Button:

def \_\_init\_\_(self, text, x, y, color):

self.text = text

self.x = x

self.y = y

self.color = color

self.width = 150

self.height = 100

def draw(self, win):

pygame.draw.rect(win, self.color, (self.x, self.y, self.width, self.height))

font = pygame.font.SysFont("comicsans", 40)

text = font.render(self.text, 1, (255,255,255))

win.blit(text, (self.x + round(self.width/2) - round(text.get\_width()/2), self.y + round(self.height/2) - round(text.get\_height()/2)))

def click(self, pos):

x1 = pos[0]

y1 = pos[1]

if self.x <= x1 <= self.x + self.width and self.y <= y1 <= self.y + self.height:

return True

else:

return False

def redrawWindow(win, game, p):

win.fill((128,128,128))

if not(game.connected()):

font = pygame.font.SysFont("comicsans", 80)

text = font.render("Waiting for Player...", 1, (255,0,0), True)

win.blit(text, (width/2 - text.get\_width()/2, height/2 - text.get\_height()/2))

else:

font = pygame.font.SysFont("comicsans", 60)

text = font.render("Your Move", 1, (0, 255,255))

win.blit(text, (80, 200))

text = font.render("Opponents", 1, (0, 255, 255))

win.blit(text, (380, 200))

move1 = game.get\_player\_move(0)

move2 = game.get\_player\_move(1)

if game.bothWent():

text1 = font.render(move1, 1, (0,0,0))

text2 = font.render(move2, 1, (0, 0, 0))

else:

if game.p1Went and p == 0:

text1 = font.render(move1, 1, (0,0,0))

elif game.p1Went:

text1 = font.render("Locked In", 1, (0, 0, 0))

else:

text1 = font.render("Waiting...", 1, (0, 0, 0))

if game.p2Went and p == 1:

text2 = font.render(move2, 1, (0,0,0))

elif game.p2Went:

text2 = font.render("Locked In", 1, (0, 0, 0))

else:

text2 = font.render("Waiting...", 1, (0, 0, 0))

if p == 1:

win.blit(text2, (100, 350))

win.blit(text1, (400, 350))

else:

win.blit(text1, (100, 350))

win.blit(text2, (400, 350))

for btn in btns:

btn.draw(win)

pygame.display.update()

btns = [Button("Rock", 50, 500, (0,0,0)), Button("Scissors", 250, 500, (255,0,0)), Button("Paper", 450, 500, (0,255,0))]

def main():

run = True

n = Network()

player = int(n.getP())

print("You are player", player)

while run:

try:

game = n.send("get")

except:

run = False

print("Couldn't get game")

break

if game.bothWent():

redrawWindow(win, game, player)

pygame.time.delay(500)

try:

game = n.send("reset")

except:

run = False

print("Couldn't get game")

break

font = pygame.font.SysFont("comicsans", 90)

if (game.winner() == 1 and player == 1) or (game.winner() == 0 and player == 0):

text = font.render("You Won!", 1, (255,0,0))

elif game.winner() == -1:

text = font.render("Tie Game!", 1, (255,0,0))

else:

text = font.render("You Lost...", 1, (255, 0, 0))

win.blit(text, (width/2 - text.get\_width()/2, height/2 - text.get\_height()/2))

pygame.display.update()

pygame.time.delay(2000)

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.quit()

if event.type == pygame.MOUSEBUTTONDOWN:

pos = pygame.mouse.get\_pos()

for btn in btns:

if btn.click(pos) and game.connected():

if player == 0:

if not game.p1Went:

n.send(btn.text)

else:

if not game.p2Went:

n.send(btn.text)

redrawWindow(win, game, player)

def menu\_screen():

run = True

while run:

win.fill((128, 128, 128))

font = pygame.font.SysFont("comicsans", 60)

text = font.render("Click to Play!", 1, (255,0,0))

win.blit(text, (100,200))

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

run = False

if event.type == pygame.MOUSEBUTTONDOWN:

run = False

main()

while True:

menu\_screen()