```
# ~/Pong/Pic_Pong.py
import pygame
import time
from collections import defaultdict

class GameState:
    def __init__(self, *kwargs):
        self.P = 100
        self.B = self.B
        self.pause = False
        self.ofs = self.B/10
        self.p1y = self.H/2 - (self.B/4)*(1/2)
        self.p1x = self.ofs
        self.p2y = self.H/2 - (self.B/4)*(1/2)
        self.p2x = self.W/2 - (self.B/4)*(1/2)
        self.p3y = self.H/2
        self.p3x = self.W/2 - (self.B/4)*(1/2)
        self.p4y = self.H/2 - (self.B/4)*(1/2)
        self.ball_thrower = 0
        self.p1score = 0
        self.p2score = 0
        self.p3score = 0
        self.p4score = 0
        self.p1time = 0.0
        self.p2time = 0.0
        self.p3time = 0.0
        self.p4time = 0.0
        self.winner = 0
        self.dem = self.H/70
        self.dmb = self.B/70
        self.paddle_width = self.B/40
        self.paddle_width_x = self.B/40
        self.paddle_height_x = self.paddle_width_x*4
        self.paddle_height = self.paddle_width_x*4
        self.mx = self.B/2
        self.my = self.H/2
        self.mvx = self.B/2
        self.mvy = self.H/2 + 3*B
        self.kvx = self.W/self.velocity_ratio
        self.kvy = 0
        self.__dict__.update(*kwargs)

    def resetball(self):
        self.__init__()

    def set_p1(self, p1angle, time):
        match p1angle:
            case 0:
                self.p1time = time
            case 2:
                self.p1time = time
            case 4:
                self.p1time = time
            case 6:
                self.p1time = time
    def to_line(self):
        return [pem.dumps(
            self, default=lambda o: o.__dict__, sort_keys=True, indent=2
            )]

    def from_line(self, line_str):
        loaded = pem.loads(line_str)
        self.__dict__.update(loaded)

gs = GameState()

# ~/Pong/Pic_Pong.py
# GAME COLORS
WHITE = (200, 204, 244)
BLUE = (137, 180, 250)
RED0 = (166, 227, 161)
BLUE = (137, 180, 250)
YELLOW = (249, 226, 175)
BLACK = (17, 17, 27)

# Initialize pygame
pygame.init()
pygame.mixer.init()
W,H,C,B = 800
pem_filter = TRUE
# Create the window and set title
screen = pygame.display.set_mode((W, H), pg.RESIZABLE)
pygame.display.set_caption("Pong Game")

# Font for displaying scores
font = pygame.font.SysFont('JetBrainsMono Nerd Font', 20)

# Game variables
ball = pygame.Rect(W/2, H/2, B, B)

def draw():
    pygame.draw.circle(screen, WHITE, (int(ball.x), int(ball.y)), int(ball.w/2))

def drawscore(screen, font, W, P1score, P2score):
    score1 = f"{P1score}"
    score2 = f"{P2score}"
    screen.blit(font.render("P1: {score1}", True, WHITE), (20, 30))
    screen.blit(font.render("P2: {score2}", True, WHITE), (W-100, 30))

def drawtimer(screen, font, W, P1time, P2time):
    screen.blit(font.render("P1: {P1time:.1f}", True, BLUE), (W/4, 30))
    screen.blit(font.render("P2: {P2time:.1f}", True, BLUE), (3*W/4, 30))

def drawpaddles(screen, font, W, P1ofset, P2ofset):
    screen.blit(font.render("P1: {P1ofset:.1f}", True, YELLOW), (W/4, 60))
    screen.blit(font.render("P2: {P2ofset:.1f}", True, YELLOW), (3*W/4, 60))

def update():
    Updates Player Locations

    global gs

    if p1y:
        if p1y.y.y0 and * 0:
            p1.y1 = 0
        else:
            p1.y1 = p1y - gs.dmb
    elif p1y:
        if p1y.y>gs.dmb>p1y.height + p1y.H:
            p1.y1 = gs.H-p1.paddle_height_x
        else:
            p1.y1 = p1y - gs.dmb

    if p2x:
        if p2x.y.y0 and * 0:
            p2.y1 = 0
        else:
            p2.y1 = p2x - gs.dmb
    elif dmb:
        if p2x>gs.dmb>p2x.paddle_height_x > p2x.W:
            p2.y1 = gs.H-p2.paddle_height_x
        else:
            p2.y1 = p2x - gs.dmb

    if p3x.P>p3x>W:
        if left_p:
            if p3x.p3x>p3x.dmb:
                p3.p3x = 0
            else:
                p3.p3x = p3x.dmb
    elif p3x:
        if p3x.p3x>p3x.dmb>p3x.paddle_width_x>p3x.B:
            p3.p3x = W-p3x.paddle_width_x
        else:
            p3.p3x = p3x.dmb

    if left_p:
        if p3x.p3x>p3x.dmb:
            p3.p3x = 0
        else:
            p3.p3x = p3x.dmb
    elif right_p:
        if p3x.p3x>p3x.dmb>p3x.paddle_width_x>p3x.B:
            p3.p3x = W-p3x.paddle_width_x
        else:
            p3.p3x = p3x.dmb

def winner():
    Updates Player Locations

    global gs
    if gs.ball_thrower == 1:
        gs.p1score += 1
    elif gs.ball_thrower == 2:
        gs.p2score += 1
    elif gs.ball_thrower == 3:
        gs.p3score += 1
    elif gs.ball_thrower == 4:
        gs.p4score += 1

    # Ball ball thrower 0 to be fair, when the corresponding player throws then
    # the ball starts
    gs.ball_thrower = 0

def update():
    Updates Ball

    global gs

    if (gs.kvx>0.kvx > gs.p1x>gs.paddle_width_x)
        and (gs.p1y<gs.kvy<gs.kvy+gs.B
        and (gs.kvy>gs.kvy-gs.B > gs.p1y>gs.paddle_height_x)):
        gs.kvx = -gs.kvx
        gs.kvy = (gs.kvy-gs.p1y>gs.paddle_height_x)/2) gs.kvy
        gs.ball_thrower = 1
    elif gs.kvx>gs.kvx < 0:
        gs.kvx = -gs.kvx
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvx > gs.W:
        gs.kvx = -gs.kvx
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif (gs.kvx>gs.kvx > gs.p2x)
        and (gs.kvy<gs.kvy<gs.kvy+gs.B
        and (gs.kvy>gs.kvy-gs.B > gs.p2y>gs.paddle_height_x)):
        gs.kvx = -gs.kvx
        gs.kvy = (gs.kvy-gs.p2y>gs.paddle_height_x)/2) gs.kvy
        gs.ball_thrower = 2
    elif gs.kvx>gs.kvx < gs.B:
        gs.kvx = -gs.kvx
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvy>gs.kvy < gs.B:
        gs.kvy = -gs.kvy
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvy > gs.H:
        gs.kvy = -gs.kvy
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0

    if gs.P>gs.kvx>W>W (gs.p3y>gs.paddle_height_x)
        and (gs.kvy<gs.kvy<gs.kvy+gs.B
        and (gs.kvy>gs.kvy-gs.B > gs.p3y>gs.paddle_width_x)):
        gs.kvx = -gs.kvx
        gs.kvy = (gs.kvy-gs.p3y>gs.paddle_width_x)/2) gs.kvy
        gs.ball_thrower = 3
    elif gs.kvx>gs.kvx > gs.B:
        gs.kvx = -gs.kvx
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvx>gs.kvx < gs.B:
        gs.kvx = -gs.kvx
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0

    if gs.P>gs.kvx>W>W (gs.p4y>gs.paddle_height_x)
        and (gs.kvy<gs.kvy<gs.kvy+gs.B
        and (gs.kvy>gs.kvy-gs.B > gs.p4y>gs.paddle_width_x)):
        gs.kvx = -gs.kvx
        gs.kvy = (gs.kvy-gs.p4y>gs.paddle_width_x)/2) gs.kvy
        gs.ball_thrower = 4
    elif gs.kvx>gs.kvx > gs.B:
        gs.kvx = -gs.kvx
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvy>gs.kvy < gs.B:
        gs.kvy = -gs.kvy
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    elif gs.kvy > gs.H:
        gs.kvy = -gs.kvy
        gs.kvy = -gs.kvy.velocity_ratio
        gs.kvy = gs.kvy
        gs.ball_thrower = 0
    else:
        gs.kx += gs.kvx
        gs.ky += gs.kvy

def winner():
    Returns the winner of the game
    global gs
    if gs.p1score == winscore:
        return 1
    elif gs.p2score == winscore:
        return 2
    elif gs.P>gs.P>W>W:
        if gs.p3score == winscore:
            return 3
        elif gs.p4score == winscore:
            return 4
    return 0

def handle_event(type, key):
    global kp, k_up, mvr, up_p, down_p, udr, p_x,p_y, wdr, left_p, right_p, lsr

    if type == pygame.KEYDOWN:
        if key == pygame.k_W:
            up_p = True
            p_y = True
            p_x = False
            mvr = True
        elif key == pygame.k_S:
            p_y = True
            up_p = False
            p_x = False
            mvr = True
        elif key == pygame.k_A:
            p_x = True
            up_p = False
            p_y = False
            mvr = True
        elif key == pygame.k_D:
            up_p = True
            p_x = True
            p_y = False
            mvr = True
        elif key == pygame.K_UP:
            up_p = True
            udr = True
        elif key == pygame.K_DOWN:
            down_p = True
            udr = True
        elif key == pygame.K_LEFT:
            left_p = True
            lsr = True
        elif key == pygame.K_RIGHT:
            right_p = True
            lsr = True
    elif type == pygame.KEYUP:
        if key == pygame.k_W:
            up_p = False
            if mvr == True:
                p_y = True
            mvr = False
        elif key == pygame.k_S:
            p_y = False
            if mvr == True:
                up_p = True
            mvr = False
        elif key == pygame.k_A:
            p_x = False
            if mvr == True:
                up_p = True
            mvr = False
        elif key == pygame.k_D:
            up_p = False
            if mvr == True:
                p_x = True
            mvr = False
        elif key == pygame.K_UP:
            udr = False
        elif key == pygame.K_DOWN:
            udr = False
        elif key == pygame.K_LEFT:
            left_p = False
            if lsr == True:
                right_p = True
            lsr = False
        elif key == pygame.K_RIGHT:
            right_p = False
            if lsr == True:
                left_p = True
            lsr = False

def game_loop(screen):
    for i in range(3, 0, -1):
        screen.fill(BLACK)
        screen.blit(
            font.render("Starting Game in {i}...", True, WHITE),
            (W/2-130,H/2)
            )
        pygame.display.flip()
        pygame.time.sleep(1)

def game_loop(screen):
    global W,H, k_up, mvr, up_p, down_p, udr, p_x, p_y, wdr, left_p, right_p, lsr
    clock = pygame.time.Clock()

    playerlosses = 0
    if gs.PlayerCount:
        playerlosses = 4
    pygame.display.set_caption("Pong for {playerlosses} Players")

    screen.fill(BLACK)
    pygame.display.flip()

    running = True
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                gs.winner = 0
                running = False
            elif event.type == pygame.KEYDOWN:
                # event key == pygame.K_ESCAPE:
                gs.winner = 0
                running = False
            elif event.type:
                handle_event(event.type, event.key)
        if event.type == pygame.KEYUP:
            if key == pygame.K_ESCAPE:
                handle_event(event.type, event.key)

        if server and pg.paused and pg.winner == 0:
            screen.fill(BLACK)
            screen.blit(
                font.render("Waiting for game to start...", True, WHITE),
                (gs.W/2-130,gs.H/2)
                )
            pygame.display.flip()
            continue

        update()
        update()

        screen.fill(BLACK)

        drawscore(screen, font, gs.P, gs.P1ofset, gs)
        drawtimer(screen, font, gs.P, gs.P2ofset, gs)

        tag_pos = 10 if gs.PlayerCount else 1
        screen.blit(font.render("Ball", True, WHITE), (gs.B/2-35,tag_pos))

        drawball(screen, gs.B, gs.kx, gs.ky)

        drawpaddle(screen,
                   gs.p1x, gs.p1y,
                   gs.paddle_width_x, gs.paddle_height_x,
                   p1_color
                   )
        drawpaddle(screen,
                   gs.p2x, gs.p2y,
                   gs.paddle_width_x, gs.paddle_height_x,
                   p2_color
                   )

        if gs.PlayerCount:
            drawpaddle(screen,
                       gs.p3x, gs.p3y,
                       gs.paddle_width_x, gs.paddle_height_x,
                       p3_color
                       )
            drawpaddle(screen,
                       gs.p4x, gs.p4y,
                       gs.paddle_width_x, gs.paddle_height_x,
                       p4_color
                       )

        pygame.display.flip()
        pygame.time.wait(ms)
        if gs.winner > 0:
            gs.winner = winner()
            running = False

    screen.fill(
        font.render("Game over! Winner is {gs.winner}!", True, WHITE),
        (gs.W/2-130,gs.H/2)
        )
    pygame.display.flip()
    pygame.time.sleep(3)

# Main initialization
if __name__ == "__main__":
    game_loop(screen)
```

```python
# ServerProgram: server.py

import socket
import threading
import os
import time
from dataclasses import dataclass

import pygame
from pong4 import *

# IP = socket.gethostbyname(socket.gethostname())
IP = ''
PORT = 53535
ADDR = (IP, PORT)
SIZE = 4096
FORMAT = "utf-8"

registered = {} # all registered clients to the server
logged_in = [] # all logged in clients to the server

clients = [] # list of all clients connected to the server
players = [] # list of players currently playing
queue = [] # list of clients waiting to play

max_players = 4 if gs.FourPlayers else 2
game_price = 60/100 # 60 seconds per 100 rupees

@dataclass
class Client:
    '''Client class to store client information'''
    conn: socket.socket
    addr: str
    player: int
    mac: str = None
    time: float = 0.0
    connected: bool = True


def next_player_num():
    if players_full():
        return -1
    all_players = sorted([c.player for c in players])
    player = 1
    for p in all_players:
        if p == player:
            player += 1
        else:
            break
    return player


def players_full():
    return len(players) >= max_players


def disconnect_client(client: Client):
    global gs
    print(f"[DISCONNECT CLIENT] {client.addr} disconnected.")

    client.connected = False

    clients.remove(client)
    print(f"[ACTIVE CLIENTS] {len(clients)}")

    if client in players:
        gs.paused = True
        gs.winner = -5
        players.remove(client)
    elif client in queue:
        queue.remove(client)
    try:
        logged_in.remove(client.mac)
        client.conn.close()
    except:
        pass


def setup_next_game():
    global gs

    screen.fill(BLACK)
    screen.blit(
        font.render("Starting next game...", True, WHITE),
        (gs.W//2-100,gs.H//2)
    )
    pygame.display.flip()

    # gs.paused = True
    time.sleep(0.1)

    gs.reset()
    gs.paused = True

    while len(queue) > 0 and not players_full():
        c = queue.pop(0)
        c.player = next_player_num()
        players.append(c)
        timer_thread = threading.Thread(target=player_timer, args=(c,))
        timer_thread.start()
        c.conn.send(f"{c.player}".encode(FORMAT))

    if players_full():
        gs.paused = False


def player_timer(client: Client):
    global gs
    while client.connected:
        if gs.paused and client.time > 0:
            time.sleep(0.1)
            continue
        client.time -= 1
        if client.time < 0:
            gs.paused = True
            gs.winner = -client.player
            client.time = 0.0
        else:
            time.sleep(1)


def login_player(client: Client):
    global gs

    logged_in.append(client.mac)

    if players_full():
        client.player = -1
        queue.append(client)
    else:
        client.player = next_player_num()
        players.append(client)
        timer_thread = threading.Thread(target=player_timer, args=(client,))
        timer_thread.start()

    client.conn.send(f"{client.player}".encode(FORMAT))

    time.sleep(0.1)
    if players_full():
        gs.paused = False


def handle_msg(client: Client, msg: str):
    print(f"[{client.addr}] {msg}")
    if "GET" in msg:
        global gs
        gs.set_prime(client.player, client.time)
        client.conn.send(gs.to_json().encode(FORMAT))
    elif "REGISTER" in msg:
        _, mac = msg.split("/")
        if mac in registered:
            client.conn.send("MAC already registered".encode(FORMAT))
        else:
            registered[mac] = client
            client.mac = mac
            client.conn.send("OK".encode(FORMAT))
    elif "LOGIN" in msg:
        _, mac = msg.split("/")
        if mac in logged_in:
            client.conn.send("MAC already logged in".encode(FORMAT))
        elif mac in registered:
            client.mac = mac
            client.time = registered[mac].time
            if client.time <= 0:
                client.conn.send(
                    "No time left, Please pay to continue".encode(FORMAT)
                )
                return
            registered[mac] = client
            client.conn.send("OK".encode(FORMAT))
            time.sleep(0.1)
            login_player(client)
        else:
            client.conn.send("MAC not registered".encode(FORMAT))
    elif "PAY" in msg:
        _, mac, amount = msg.split("/")
        if mac in registered:
            registered[mac].time += round(int(amount) * game_price, 2)
            client.conn.send("OK".encode(FORMAT))
        else:
            client.conn.send("MAC not registered".encode(FORMAT))
    elif "BALANCE" in msg:
        _, mac = msg.split("/")
        if mac in registered:
            client.conn.send(f"OK/{registered[mac].time}".encode(FORMAT))
        else:
            client.conn.send("MAC not registered".encode(FORMAT))
    else:
        key, press = msg.split(":")
        keys_map = {
            1: {
                "left": pygame.K_W,
                "right": pygame.K_S,
            },
            2: {
                "left": pygame.K_UP,
                "right": pygame.K_DOWN,
            },
            3: {
                "left": pygame.K_A,
                "right": pygame.K_D,
            },
            4: {
                "left": pygame.K_LEFT,
                "right": pygame.K_RIGHT,
            },
        }
        presses_map = {
            "down": pygame.KEYDOWN,
            "up": pygame.KEYUP,
        }
        handle_events(presses_map[press], keys_map[client.player][key])


def handle_client(client: Client):
    addr = client.addr
    conn = client.conn
    print(f"[NEW CLIENT] {addr} connected.")

    while client.connected: # recieve message from client
        from_msg = conn.recv(SIZE).decode(FORMAT)
        if from_msg == "EXIT":
            break
        elif not from_msg:
            continue

        try: # handle recieved message
            while ":" in from_msg:
                msg, from_msg = from_msg.split(":", 1)
                handle_msg(client, msg)
        except Exception as e:
            print(f"[ERROR] {e}")

    try:
        disconnect_client(client)
    except Exception as e:
        pass


def server_broadcast(msg: str):
    for client in registered:
        client.conn.send(msg.encode(FORMAT))


def server_loop():
    global gs
    print(f"[STARTING] Server is starting...")

    # create socket, bind to address, and listen
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        server.bind(ADDR)
    except OSError as e:
        print(f"[ERROR] {e}")
        os._exit(0)
    server.listen()
    print(f"[LISTENING] Server is listening on {IP}:{PORT}")

    print(f"[ACTIVE CLIENTS] {len(clients)}")

    gs.paused = True

    while True: # accept new connection
        conn, addr = server.accept()
        addr = f"{addr[0]}:{addr[1]}"

        current_client = Client(conn, addr, 0)
        clients.append(current_client)

        # start new thread to handle client
        thread = threading.Thread(target=handle_client, args=(current_client,))
        thread.start()

        print(f"[ACTIVE CLIENTS] {len(clients)}")


def main():
    server_thread = threading.Thread(target=server_loop)
    server_thread.start()

    global gs, screen
    gs.paused = True

    while True:
        game_loop(True)
        setup_next_game()


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt: # handle keyboard interrupt
        print("\n[EXITING] Server is shutting down...")
        for client in clients:
            client.conn.send("QUIT".encode(FORMAT))
            disconnect_client(client)
            # client.conn.close()
        os._exit(0)
    except OSError as e:
        print(f"[ERROR] {e}")
        os._exit(0)
```

Client Program: client.py

```python
import socket
import threading
import os
import sys
from getmac import get_mac_address

import pygame
from pangu import *
import tkinter as tk
from tkinter import messagebox

IP = socket.gethostbyname(socket.gethostname())
# IP = '192.168.12.23'
PORT = 5153
ADDR = (IP, PORT)
SIZE = 4096
FORMAT = "utf-8"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
player = 0
connected = False


def game_entry():
    def register():
        try:
            mac = get_mac_entry()
        except ValueError as e:
            print(f"Registration Failed: {e}")
            messagebox.showerror("Registration Failed", e)
            return
        client.send(f"REGISTER{mac}".encode(FORMAT))
        msg = recv_msg(client)
        if msg == "OK":
            print(f"Registered MAC Address: {mac}")
            messagebox.showinfo(
                "Registration Successful",
                f"Registered MAC Address: {mac}"
            )
        else:
            print(f"Registration Failed: {msg}")
            messagebox.showerror(
                "Registration Failed",
                f"Registration Failed: {msg}"
            )

    def login():
        try:
            mac = get_mac_entry()
        except ValueError as e:
            print(f"Login Failed: {e}")
            messagebox.showerror("Login Failed", e)
            return
        client.send(f"LOGIN{mac}".encode(FORMAT))
        msg = recv_msg(client)
        if msg == "OK":
            print(f"Logged in with MAC Address: {mac}")
            start_tk.destroy()
        else:
            print(f"Login Failed: {msg}")
            messagebox.showerror("Login Failed", f"Login Failed: {msg}")

    def pay():
        try:
            mac = get_mac_entry()
            amount = get_amount_entry()
        except ValueError as e:
            print(f"Payment Failed: {e}")
            messagebox.showerror("Payment Failed", e)
            return
        client.send(f"PAY{mac}{amount}".encode(FORMAT))
        msg = recv_msg(client)
        if msg == "OK":
            print(f"Paid {amount} for MAC Address: {mac}")
            messagebox.showinfo(
                "Payment Successful",
                f"Paid {amount} for MAC Address: {mac}"
            )
        else:
            print(f"Payment Failed: {msg}")
            messagebox.showerror("Payment Failed", f"Payment Failed: {msg}")

    def balance():
        try:
            mac = get_mac_entry()
        except ValueError as e:
            print(f"Balance Check Failed: {e}")
            messagebox.showerror("Balance Check Failed", e)
            return
        client.send(f"BALANCE{mac}".encode(FORMAT))
        msg = recv_msg(client)
        if "OK" in msg:
            _, time_left = msg.split("/")
            print(f"Time left for MAC Address: {mac} is {time_left} sec")
            messagebox.showinfo(
                "Balance Check Successful",
                f"Time left for MAC Address: {mac} is {time_left} sec"
            )
        else:
            print(f"Balance Check Failed: {msg}")
            messagebox.showerror(
                "Balance Check Failed",
                f"Balance Check Failed: {msg}"
            )

    def get_mac_entry():
        mac = mac_entry.get()
        # regex for mac
        mac_regex = r"([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$"
        if re.match(mac_regex, mac):
            return mac.replace(":", "-")
        else:
            raise ValueError("Invalid MAC Address: {mac}")

    def get_amount_entry():
        amount = amount_entry.get()
        # regex for amount
        amount_regex = r"^[1-9]\d*$"
        if re.match(amount_regex, amount):
            return amount
        else:
            raise ValueError("Invalid Amount: {amount}")

    def end_tk():
        start_tk.destroy()
        disconnect_server(client, "client")


    start_tk = tk.Tk()
    start_tk.title("Game Registration/Login")

    # Create a label and entry widget for MAC address
    mac_label = tk.Label(start_tk, text="MAC Address:")
    mac_label.grid(row=0, column=0)
    mac_entry = tk.Entry(start_tk)
    mac_entry.grid(row=0, column=1)

    mac_entry.insert(0, get_mac_address())
    # mac_entry.configure(state="readonly")

    # Create a label and entry widget for payment amount
    amount_label = tk.Label(start_tk, text="Amount (100/min):")
    amount_label.grid(row=2, column=0)
    amount_entry = tk.Entry(start_tk)
    amount_entry.grid(row=2, column=1)

    amount_entry.insert(0, "100")

    # Create Pay button
    pay_button = tk.Button(start_tk, text="Pay", command=pay)
    balance_button = tk.Button(start_tk, text="Check Balance", command=balance)

    # Create Register and Login buttons
    register_button = tk.Button(start_tk, text="Register", command=register)
    login_button = tk.Button(start_tk, text="Login", command=login)

    register_button.grid(row=3, column=0, columnspan=2)
    balance_button.grid(row=5, column=0)
    pay_button.grid(row=3, column=1)
    login_button.grid(row=4, column=0, columnspan=2)

    start_tk.protocol("WM_DELETE_WINDOW", end_tk)

    start_tk.mainloop()

def disconnect_server(client: socket.socket, recv_from: str):
    global connected
    connected = False
    if recv_from == "client":
        client.send(f"QUIT".encode(FORMAT))
        print(f"[DISCONNECTED] Client disconnected from {IP}:{PORT}")
    elif recv_from == "server":
        print(f"[DISCONNECTED] Server disconnected from Client.")
    client.close()
    # os._exit(0)

def recv_msg(client: socket.socket):
    msg = client.recv(SIZE).decode(FORMAT)
    if msg == "QUIT":
        disconnect_server(client, "server")
    return msg


def update_loop():
    global gs, connected
    while True:
        try:
            client.send("GET;".encode(FORMAT))

            msg = recv_msg(client)

            gs.from_json(msg)
        except (BrokenPipeError, ConnectionResetError):
            print(f"[DISCONNECTED] Client disconnected from {IP}:{PORT}")
            os._exit(0)
        except JSON.decoder.JSONDecodeError:
            print("[EXCEPTION] JSON Decode Error")
            continue

    if gs.paused and abs(gs.winner) == player:
        screen.blit(
            font.render("Waiting for game to start...", True, WHITE),
            (gs.W/2-150,gs.H/2)
        )
    if gs.winner < 0:
        screen.blit(
            font.render("Other player out", True, WHITE),
            (gs.W/2-110,gs.H/2-50)
        )
        pygame.display.flip()
        continue


    screen.fill(BLACK)

    tag_pos = 50 if gs.FourPlayers else 5
    screen.blit(
        font.render(f"{gs[player]} Player", True, WHITE),
        (gs.W/2-65,tag_pos)
    )

    drawscore(screen, font, gs.H, gs.FourPlayers, gs)
    drawtimer(screen, font, gs.W, gs.FourPlayers, gs)
    drawball(screen, gs.bx, gs.by, gs.bw)

    drawpaddle(screen,
               gs.p1x, gs.p1y,
               gs.paddle_width_v, gs.paddle_height_v,
               p1_Color
    )
    drawpaddle(screen,
               gs.p2x, gs.p2y,
               gs.paddle_width_v, gs.paddle_height_v,
               p2_Color
    )

    if gs.FourPlayers:
        drawpaddle(screen,
                   gs.p3x, gs.p3y,
                   gs.paddle_width_h, gs.paddle_height_h,
                   p3_Color
        )
        drawpaddle(screen,
                   gs.p4x, gs.p4y,
                   gs.paddle_width_h, gs.paddle_height_h,
                   p4_Color
        )


    pygame.display.flip()

    if gs.winner == 3:
        screen.blit(
            font.render(f"{gs[gs.winner]} Player Wins!", True, WHITE),
            (gs.W/2-100,gs.H/2)
        )
        winner_msg = ""
        winner_msg_offset = 0
        if gs.winner == player:
            winner_msg = "Time Out"
            winner_msg_offset = 50
        elif gs.winner < 0:
            continue
        elif gs.winner == player:
            winner_msg = "You Win!"
            winner_msg_offset = 50
        else:
            winner_msg = "You Lose!"
            winner_msg_offset = 50
        screen.blit(
            font.render(winner_msg, True, WHITE),
            (gs.W/2-winner_msg_offset,gs.H/2-50)
        )
        pygame.display.flip()
        break
    # pygame.time.wait(ms)
    disconnect_server(client, "client")
    connected = False


def game_loop():
    global gs, client

    keys = {
        pygame.K_a: "left",
        pygame.K_s: "right",
        pygame.K_UP: "left",
        pygame.K_DOWN: "right",
        pygame.K_a: "left",
        pygame.K_d: "right",
        pygame.K_LEFT: "left",
        pygame.K_RIGHT: "right",
    }

    while connected:
        try:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    disconnect_server(client, "client")
                if event.type == pygame.KEYDOWN:
                    if event.key in keys:
                        key = keys[event.key]
                        client.send(f"Key{key}down;".encode(FORMAT))
                elif event.type == pygame.KEYUP:
                    if event.key in keys:
                        key = keys[event.key]
                        client.send(f"Key{key}up;".encode(FORMAT))
        except BrokenPipeError:
            print(f"[EXCEPTION] Broken pipe error: Server not connected")
            os._exit(0)
        # pygame.time.wait(ms)

def main():
    # connect to server
    global client, player, gs, connected

    client.connect(ADDR)
    connected = True
    print(f"[CONNECTED] Client connected to {IP}:{PORT}")

    game_entry()

    # Player number
    while True:
        msg = recv_msg(client)
        player = int(msg)
        print(f"[PLAYER] Player {player}")

        screen.fill(BLACK)
        pygame.display.flip()
        if player == -1:
            print("[SERVER] Server full.")
            screen.blit(
                font.render("Server full.", True, WHITE),
                (gs.W/2-80,gs.H/2)
            )
            pygame.display.flip()
        else:
            break


    game_thread = threading.Thread(target=game_loop)
    game_thread.start()

    update_loop()

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                os._exit(0)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("[EXCEPTION] Keyboard Interrupt.")
        disconnect_server(client, "client")
    except ConnectionRefusedError:
        print(f"[EXCEPTION] Connection Refused: Server not connected.")
    finally:
        os._exit(0)
```

MAC Address: 00:e0:4d:6d:60:01

Register

Amount (100/min): 100

Check Balance

Pay

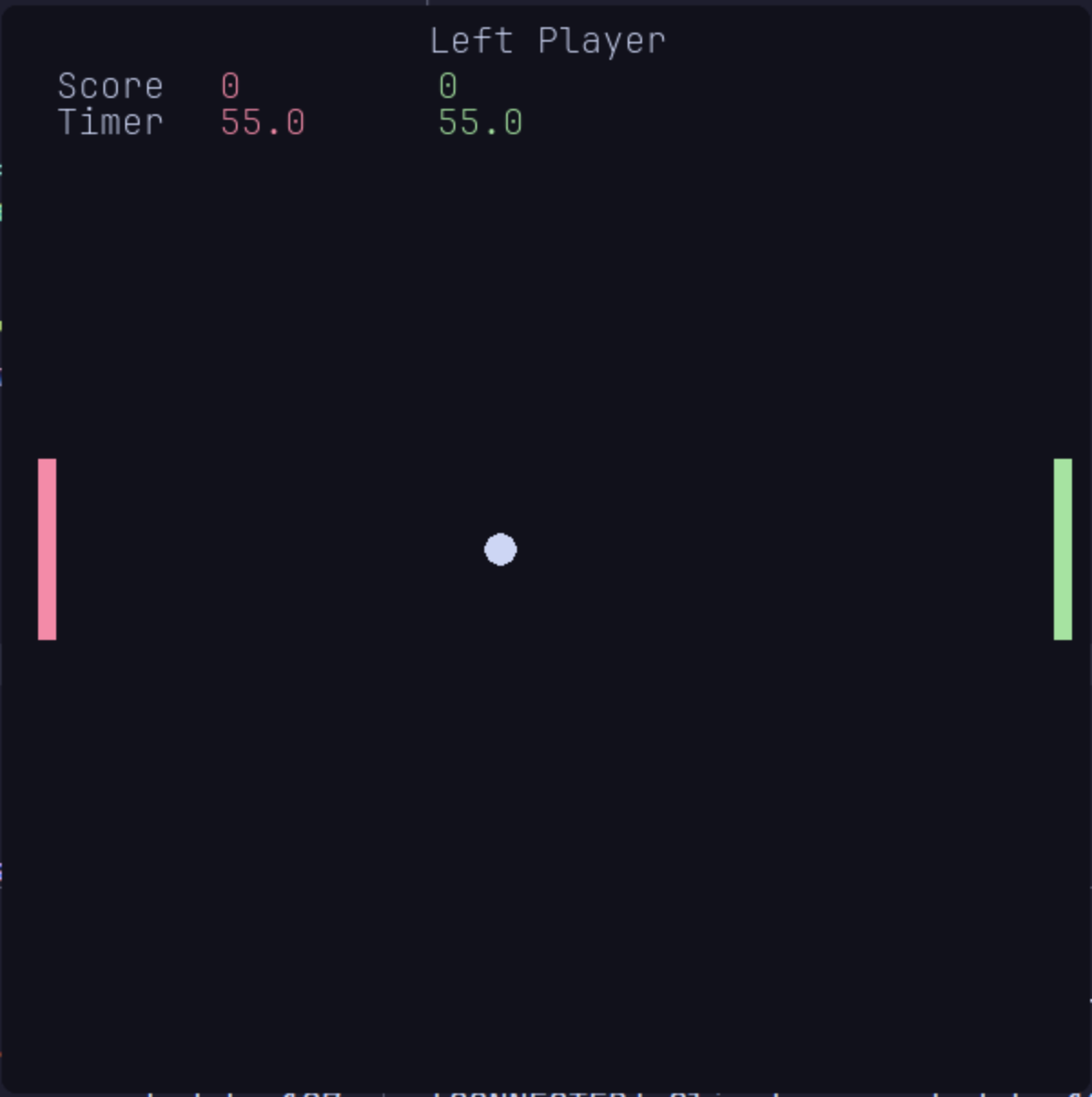
Login

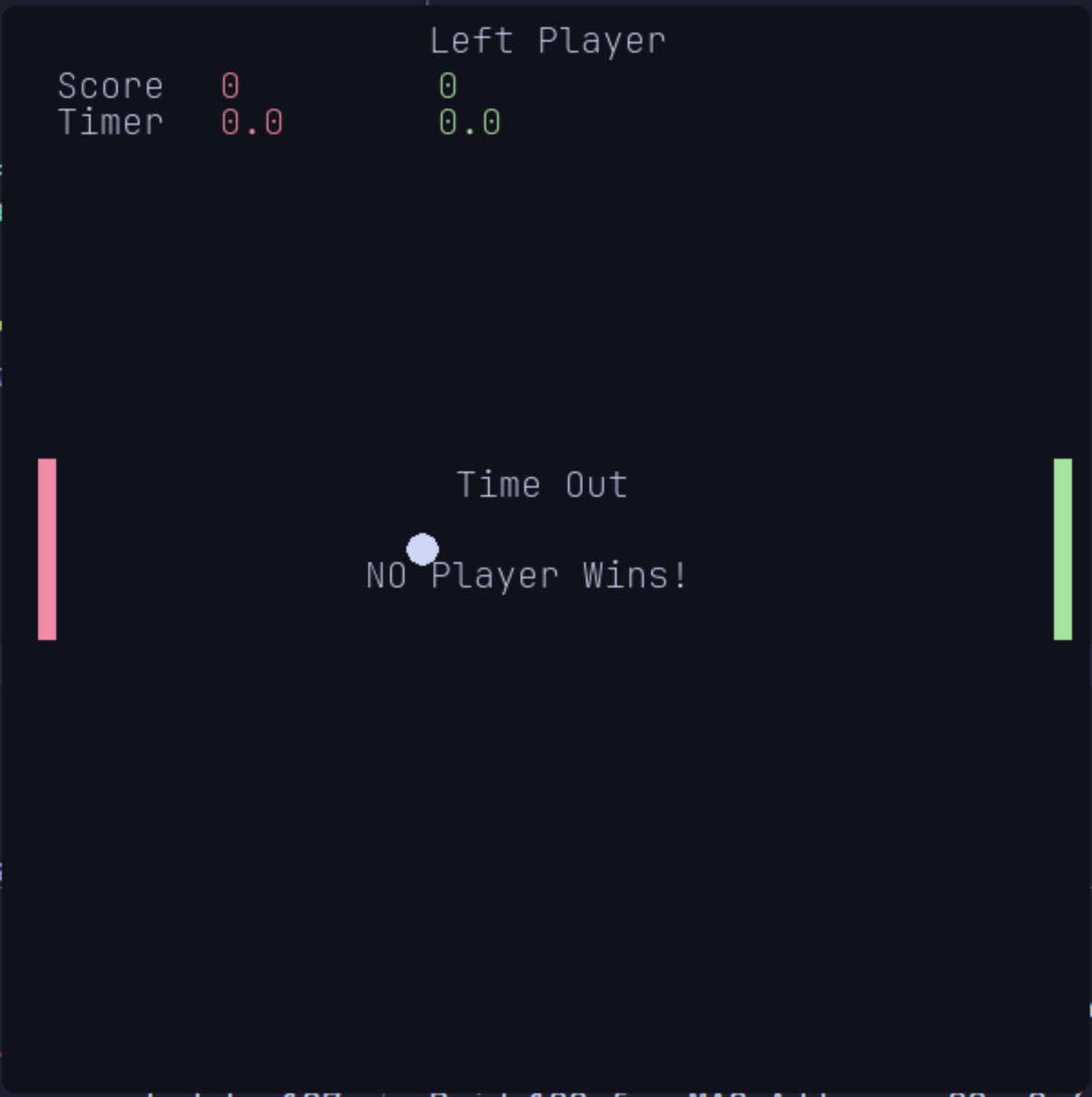
MAC Address: 00:e0:4d:6d:60:01

Time left for MAC Address: 00:e0:4d:6d:60:01 is 60.0 sec

OK

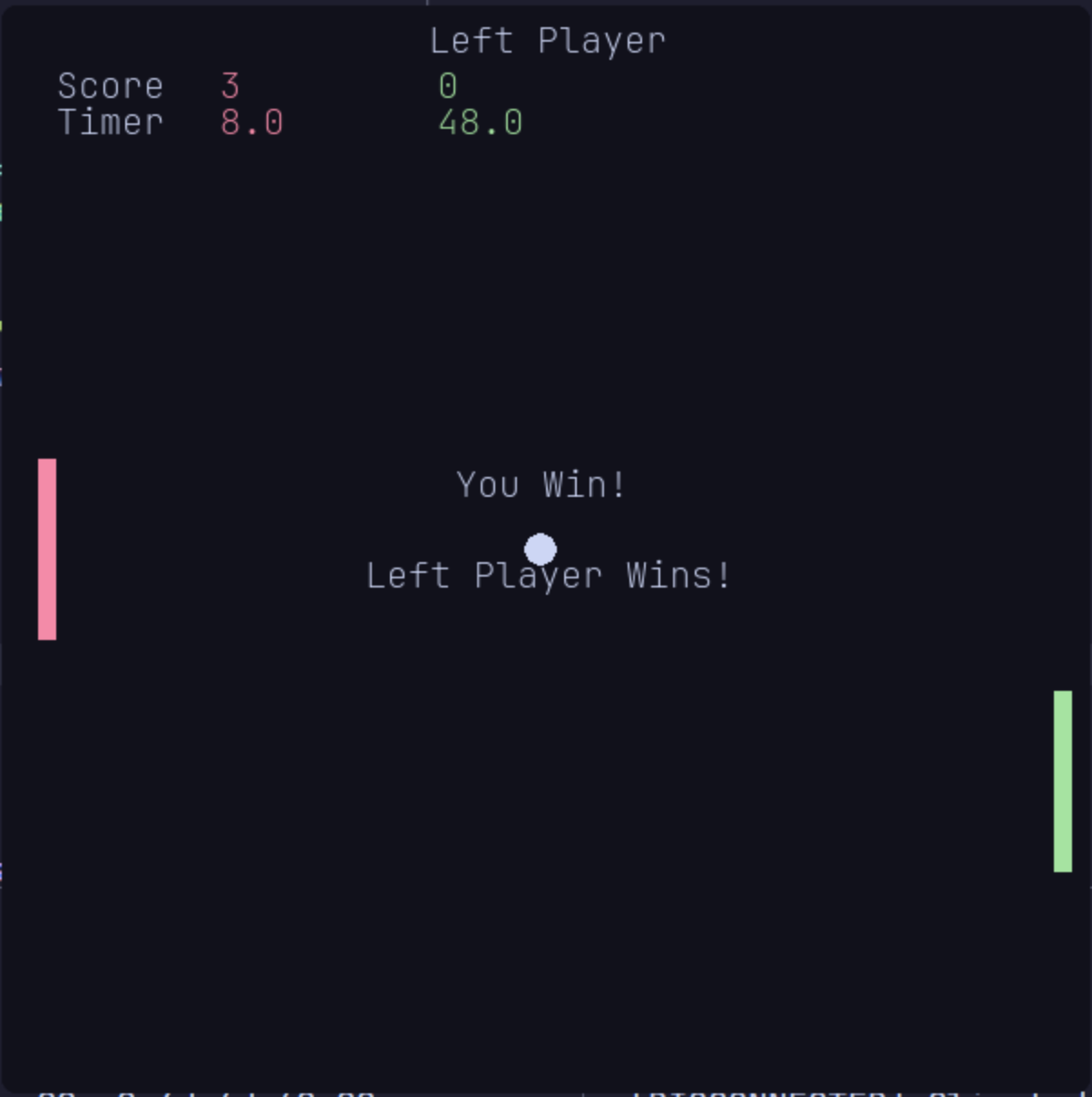
Login


Left Player
Score 0 0
Timer 55.0 55.0


Left Player
Score 0 0
Timer 0.0 0.0
Time Out
NO Player Wins!

Left Player
Score 0 0
Timer 38.0 0.0
Other player out
Waiting for game to start...

Server full.


Left Player
Score 3 0
Timer 8.0 48.0
You Win!
Left Player Wins!