Snake.py

```python
import pygame
from pygame.locals import QUIT, K_w, K_s, K_a, K_d, K_q,K_DOWN,K_UP,K_LEFT,K_RIGHT,
KEYDOWN, K_SPACE
import Entities
import regex as re
from random import randint
import threading
import socket
from time import sleep

this = None

#   Input map
input_movement = {K_w:'y-',K_s:'y+',K_d:'x+',K_a:'x-'}

#   Good colours :)
snake_colours = [
        (165, 38, 176),     #   Purple
        (240, 155, 89)      #   Brown
]

#   Deprecated
render_tick = False

#   Hard coded amount of players due to lack of time
players = {
    'P0':{'lastmovement':'-x',
        'newmovement':'-x',
        'player':Entities.Player(5,5,snake_colours[0])},
    'P1':{'lastmovement':'-x',
        'newmovement':'-x',
        'player':Entities.Player(5,10,snake_colours[1])}
}

#   IP and Port to connect to
HOST, PORT = '192.168.20.69', 9999

def quit_application():#    Close socket connection when the application is quit
    global client_socket
    client_socket.close()

def send_data(data:str):#   Easy send function
    global client_socket
    client_socket.send(data.encode('utf-8'))

def parse_data(data):#      Parsing of received data
    global this, tick, STATE_OF_APPLICATION, apple,render_tick
    tag, cmd = data.split('|')
    #   I miss switch case from Java :'(
    if tag == 'you':
        this = cmd
        return
    if tag == 'start':
        tick = 0
        STATE_OF_APPLICATION = 'GAME'
        return
    if cmd.startswith('apple'):
        apple.x, apple.y = [int(coord) for coord in cmd.split(':')[1].split(',')]
        return
    if tag == 'update':
        update_logic()
        return
    players[tag]['newmovement'] = data


def receive_data(client_socket):
    try:
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            msg = data.decode('utf-8')
            print(f'Received from server: {msg}')
            parse_data(msg)

    except Exception as e:
        print(f'Error receiving data: {e}')

def start_client():
    # Create a TCP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect to the server
    client_socket.connect((HOST, PORT))

    # Start a thread to receive data from the server
    receive_thread = threading.Thread(target=receive_data, args=(client_socket,))
    receive_thread.start()

    return client_socket

def update_logic():
    global players, last_movement, new_movement, apple
    for tag in players.keys():
        player = players[tag]
        player['lastmovement'] = player['newmovement']
        last_movement = new_movement
        player['player'].direction = player['newmovement']
        player['player'].update()
        if apple == player['player'].head:
            player['player'].eat()
            apple.new_position()
            send_data(f'apple:{apple.x},{apple.y}')


pygame.init()

#   Global  Static  Variables

TILE_SIZE:int   = 40
WIDTH:int       = 17
HEIGHT:int      = 17

SCREEN_HEIGHT = HEIGHT*TILE_SIZE
SCREEN_WIDTH = HEIGHT*TILE_SIZE
STATE_OF_APPLICATION = 'MENU'

```

```python
#   Other Vars (primarily pygame related)

game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))

#   Good colours, defo not stolen from Googles Snake...
bg_colours = [(119, 221, 119),(106, 196, 106)]

clock = pygame.time.Clock()

client_socket = start_client()

#   Sleep to ensure that variable 'this' is set
sleep(0.05)
tick = 0
new_movement = 'x-'

#   Some hardcoded variables because yes...
apple = Entities.Apple(10,10)
pygame.display.set_caption(f'Snake MP ({this}) connected to ({HOST}:{PORT})')

new_movement = '-x'
last_movement = '-x'

def game_loop_logic():
    global tick, new_movement,last_movement, apple, clock, players, render_tick
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()

        elif event.type == KEYDOWN:
            if event.key == K_q:
                pygame.quit()
            #   Check pressed key against input map
            if event.key in input_movement.keys():
                new_movement = input_movement[event.key]
                if not re.sub('[-+]','',last_movement) in new_movement:
                    send_data(new_movement)
                    pass
                else:
                    new_movement = last_movement

    try:
        pygame.display.update()
    except:
        print('Shutting down application...')
        return 0


    if tick > 6:
        tick = 0; render_tick = False
        game_screen.fill(bg_colours[0])

        _ = 0
        for y in range(0,17):
            for x in range(0,17):
                if _ % 2 == 0:
                    pygame.draw.rect(game_screen, bg_colours[1], pygame.Rect(x*TILE_SIZE,
y*TILE_SIZE,TILE_SIZE,TILE_SIZE))
                _ += 1

        #   Draw snakes
        for body in Entities.all_bodies:
            body.render(game_screen, pygame)

        #   Draw apple
        apple.render(game_screen, pygame)
    tick += 1
    return True

def menu_loop_logic():
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()

        elif event.type == KEYDOWN:
            if event.key == K_q:
                pygame.quit()
    try:
        pygame.display.update()
    except: #   If application has been quit this will throw an exception
            #   Then I'll know the program needs to be shut down and will return 0
        print('Shutting down application...')
        return 0
    game_screen.fill(bg_colours[0])
    return True
##abb8c3
LoopMap = {'MENU':menu_loop_logic,'GAME':game_loop_logic}
while True:
    status = LoopMap[STATE_OF_APPLICATION]()
    if not status:  #   Exit logic
        print('Ending application...')
        quit_application()
        break
    pygame.display.flip()
    clock.tick(60)
```