```
import random
import socket
from _thread import *
import pickle
import pygame

window_width = 320
window_height = 240
player_width = 20
player_height = 20
player_movement_speed = 20
player1_start_x = 10
player1_start_y = 120
player2_start_x = 290
player2_start_y = 120

data_size = 4096 * 2

game_speed = 30

class Bullet:
    def __init__(self, player_id,x,y,vx,vy):
        self.color = (0,0,255)
        self.width = 20 if vx != 0 else 5
        self.height = 20 if vx == 0 else 5
        self.x = x
        self.y = y
        self.vx = vx
        self.vy = vy
        self.player_id = player_id

    def draw(self, window):
        pygame.draw.rect(window,self.color,(self.x-self.width/2,
self.y-self.height/2, self.width, self.height))


class ShootDTO:
    def __init__(self):
        self.game_id = 0
        self.player_id = 0
        self.player_x = []
        self.player_y = []
        self.start_play = False
        self.msg = ''
        self.points = [0, 0]
        self.bullets0 = []
        self.bullets1 = []


class Game:
    game_id = 0
    player_ids = []
    game_dto = ShootDTO()

    def __init__(self):
        self.game_id = 0
        self.player_ids = []
        self.game_dto = ShootDTO()

    def initiate_dto(self):
        self.game_dto.player_x = [player1_start_x, player2_start_x]
        self.game_dto.player_y = [player1_start_y, player2_start_y]
        self.game_dto.game_id = self.game_id
        self.game_dto.start_play = False
        self.game_dto.points = [0, 0]


def get_dto_for_game(game_id):
    for game in game_ids:
        if game_id == game.game_id:
            return game.game_dto


def get_game(game_id):
    for game in game_ids:
        if game_id == game.game_id:
            return game


def update_game_state(dto):
    game_dto = get_dto_for_game(dto.game_id)
    game_dto.player_y[dto.player_id] = dto.player_y[dto.player_id]
    game_dto.player_x[dto.player_id] = dto.player_x[dto.player_id]
    if (dto.player_id == 0):
        nxtBullets0 = []
        for bullet in dto.bullets0:
            nxtX = bullet.x + bullet.vx
            nxtY = bullet.y  + bullet.vy
            if (nxtX > window_width or nxtX < 0 or nxtY < 0 or nxtY >
window_height):
                continue
            bullet.x = nxtX
            bullet.y = nxtY
            nxtBullets0.append(bullet)
        game_dto.bullets0 = nxtBullets0
    if (dto.player_id == 1):
        nxtBullets1 = []
        for bullet in dto.bullets1:
            nxtX = bullet.x + bullet.vx
            nxtY = bullet.y  + bullet.vy
            if (nxtX > window_width or nxtX < 0 or nxtY < 0 or nxtY >
window_height):
                continue
            bullet.x = nxtX
            bullet.y = nxtY
            nxtBullets1.append(bullet)
        game_dto.bullets1 = nxtBullets1

    for bullet in dto.bullets0:
        bulletRect =
pygame.Rect(bullet.x,bullet.y,bullet.width,bullet.height)
        if
bulletRect.colliderect(pygame.Rect(dto.player_x[1],dto.player_y[1],
player_height, player_width)):
            game_dto.points[0] += 1
            game_dto.player_x[0] = player1_start_x
            game_dto.player_y[0] = player1_start_y
            game_dto.player_x[1] = player2_start_x
            game_dto.player_y[1] = player2_start_y
            game_dto.bullets0 = []
            game_dto.bullets1 = []

    for bullet in dto.bullets1:
        bulletRect =
pygame.Rect(bullet.x,bullet.y,bullet.width,bullet.height)
        if
bulletRect.colliderect(pygame.Rect(dto.player_x[0],dto.player_y[0],
player_height, player_width)):
            game_dto.points[1] += 1
            game_dto.player_x[0] = player1_start_x
            game_dto.player_y[0] = player1_start_y
            game_dto.player_x[1] = player2_start_x
            game_dto.player_y[1] = player2_start_y
            game_dto.bullets0 = []
            game_dto.bullets1 = []



def get_game_player_id():
    if not game_ids:
        game = Game()
        game.game_id = 0
        game.player_ids.append(0)
        game.initiate_dto()
        game_ids.append(game)
        return 0, 0
    else:
        for i, game in enumerate(game_ids):
            if len(game.player_ids) == 1:
                player_id = list({0, 1} - set(game.player_ids))[0]
                game.player_ids.append(player_id)
                game.game_dto.start_play = True
                return i, player_id
        game_id = game_ids[-1].game_id + 1
        game = Game()
        game.game_id = game_id
        game.player_ids.append(0)
        game.initiate_dto()
        game_ids.append(game)
        return game_id, 0


def client_code_process(conn, game_id, player_id):
    send_dto = get_dto_for_game(game_id)
    send_dto.player_id = player_id
    conn.send(pickle.dumps(send_dto))

    clock = pygame.time.Clock()

    while True:
        clock.tick(game_speed)
        try:
            receive_dto = pickle.loads(conn.recv(data_size))
            if not receive_dto:
                break
            receive_dto.player_id = player_id
            if receive_dto.start_play:
                update_game_state(receive_dto)
            game_dto = get_dto_for_game(game_id)
            game_dto.player_id = player_id
            conn.sendall(pickle.dumps(game_dto))
        except Exception as e:
            break

    game = get_game(game_id)
    game.player_ids.remove(player_id)
    if len(game.player_ids) == 0:
        game_ids.remove(game)
    else:
        game.initiate_dto()
    conn.close()

server = "10.49.95.67"

port = 5555

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.bind((server, port))
except socket.error as e:
    print(e)

s.listen()

game_ids = []
while True:
    conn, addr = s.accept()
    game_id, player_id = get_game_player_id()
    start_new_thread(client_code_process, (conn, game_id, player_id))
```