```
import pickle
import random
import socket
import pygame
import time
import os
from pygame.locals import *
import math


window_width = 320
window_height = 240
player_width = 20
player_height = 20
player_movement_speed = 20

class ShootDTO:
    """This is a data transfer object containing the variables that
will be passed between server and clients."""

    # Initiate to default values
    def __init__(self):
        self.game_id = 0
        self.player_id = 0
        self.player_x = [] # x position of player {index}
        self.player_y = [] # y position of player {index}
        self.start_play = False # is true once wo players are
connected
        self.msg = ''
        self.points = [0, 0]
        self.bullets0 = [] # bullets fired by player 0
        self.bullets1 = [] # bullets fired by player 1

class Bullet:
    def __init__(self, player_id,x,y,vx,vy):
        self.color = (0,0,255)
        self.width = 5
        self.height = 5
        self.x = x
        self.y = y
        self.vx = vx
        self.vy = vy
        self.player_id = player_id # who shot the bullet

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


# Packet size for sending and receiving between server and clients
```

```
data_size = 4096 * 2

# FPS speed for the game clock
game_speed = 30
class Shooter:
    """This class moves the players' character on the screen."""

    def __init__(self,x,y,color):
        """Constructor for player"""
        self.x = x
        self.y = y
        self.color = color
        self.width = player_width
        self.height = player_height
        self.points = 0

    def draw(self,window):
        """ This method draws the shooter on the screen."""
        player_image = pygame.image.load('player.png').convert()
        player_image = pygame.transform.scale(player_image, (20,20))
        win.blit(player_image, (self.x - self.width // 2, self.y -
self.width // 2))

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

    def move(self,direction):
        """This method takes the string of the direction and updates
our coordinates as the player"""
        if (direction == "up" and self.y > (player_movement_speed /
2)):
            self.y -= player_movement_speed
        elif direction == "down" and self.y < (window_height -
player_height - (player_movement_speed / 2)):
            self.y += player_movement_speed
        elif direction == "left" and self.x > (player_movement_speed/
2):
            self.x -= player_movement_speed
        elif (direction == "right" and self.x < (window_width -
player_width - (player_movement_speed/2))):
            self.x += player_movement_speed

    def add_point(self):
        """Adds a point when you shoot someone"""
        self.points += 1


def update_player_bullets(dto):
    """This method takes the DTO as input"""
    global bullets0 , bullets1
```

```
    players[player_id].color = (255,0,0)
    players[opponent_id].color = (0,255,0)

    # Sets the coorindates of the players
    players[0].x = dto.player_x[0]
    players[0].y = dto.player_y[0]
    players[1].x = dto.player_x[1]
    players[1].y = dto.player_y[1]
    bullets0 = dto.bullets0
    bullets0 = dto.bullets1


# Create a socket for the server and client connection
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Server IP is where the server python file is running and listening
for connections
server = "10.49.95.67"
# Port is where server is listening.
port = 5555
addr = (server, port)

# Initiate client and server connection
client.connect(addr)
# Receive the data transfer object from server for first time
receive_dto = pickle.loads(client.recv(data_size))
print("You are player ", receive_dto.player_id)

# Retrieve the player id from the DTO
player_id = receive_dto.player_id
# The opponent id is the other id from set {0,1}
opponent_id = list({0, 1} - {receive_dto.player_id})[0]

# Initiate the Players
players = [Shooter(0, 0, (0, 0, 0)), Shooter(0, 0, (0, 0, 0))]

#initiate the bullets0
bullets0 = []
#initiate the bullets1
bullets1 = []
#init Client
update_player_bullets(receive_dto)

pygame.font.init()
screen = pygame.display.set_mode((window_width, window_height))
pygame.display.set_caption("Background Image")
background_image = pygame.image.load("background.png").convert()
background_image = pygame.transform.scale(background_image,
(window_width, window_height))
# Set the surface attributes for the window
win = pygame.display.set_mode((window_width, window_height))
```

```
run = True
# Get the game clock
clock = pygame.time.Clock()
WHITE = (255, 255, 255)
font = pygame.font.Font(None, 30)
buttons= { "quit": (40, 20) }

SCOREX = 160
SCOREY = 20

# Start the loop for the game

prevAction = time.time()
while run:
    clock.tick(game_speed)
    win.fill((0, 0, 0))

    vd = 6 #view distance
    view_rect_player = pygame.Rect(receive_dto.player_x[player_id]
-vd/2*player_width, receive_dto.player_y[player_id] - vd/
2*player_height, player_width * vd, player_height*vd)
    view_surface = pygame.Surface((window_width, window_height))
    blit_pos = (view_rect_player.left, view_rect_player.top)

    view_surface.blit(background_image, blit_pos, view_rect_player)
    screen.blit(view_surface, (0, 0))
    players[player_id].draw(win)

    # draw all the bullets

    for bullet in receive_dto.bullets0:
        bullet.draw(win)
    for bullet in receive_dto.bullets1:
        bullet.draw(win)

    for text, pos in buttons.items():
        text_surface= font.render(text, True, WHITE)
        rect = text_surface.get_rect(center=pos)
        screen.blit(text_surface, rect)

    scoreText = "Player: " + str(receive_dto.points[player_id]) + "
OP: " + str(receive_dto.points[opponent_id])
    text_surface = font.render(scoreText,True,WHITE)
    rect = text_surface.get_rect(center = (SCOREX,SCOREY))
    screen.blit(text_surface, rect)

    if (time.time() -prevAction > 0.1):
        prevAction = time.time()
        for event in pygame.event.get():
```

```
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            elif (event.type is MOUSEBUTTONUP):
                x,y = pygame.mouse.get_pos()
                coord_str = "x = " + str(x) + ", y = "+ str(y) + "\n"
                print(coord_str)
                # quit button hitbox
                if (y < 40 ) and (x > 20 and x < 60):
                    run = False
                    break
                magnitude = math.sqrt(x*x + y*y)
                if player_id == 0 and len(receive_dto.bullets0) < 2:
                    vx = (x - receive_dto.player_x[player_id])
                    vy = (y - receive_dto.player_y[player_id])
                    magnitude = math.sqrt(vx*vx + vy*vy) / 5
                    vx, vy = vx / magnitude, vy / magnitude

receive_dto.bullets0.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],vx,vy ))
                if player_id == 1 and len(receive_dto.bullets1) < 2:
                    vx = (x - receive_dto.player_x[player_id])
                    vy = (y - receive_dto.player_y[player_id])
                    magnitude = math.sqrt(vx*vx + vy*vy) / 5
                    vx, vy = vx / magnitude, vy / magnitude

receive_dto.bullets1.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],vx,vy))
            elif event.type == pygame.KEYDOWN :
                if event.key == pygame.K_LEFT:
                    players[player_id].move("left")
                elif event.key == pygame.K_RIGHT:
                    players[player_id].move("right")
                elif event.key == pygame.K_UP:
                    players[player_id].move("up")
                elif event.key == pygame.K_DOWN:
                    players[player_id].move("down")
                elif event.key == pygame.K_w:
                    if player_id == 0 and len(receive_dto.bullets0) <
2:

receive_dto.bullets0.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],0,-5 ))
                    if player_id == 1 and len(receive_dto.bullets1) <
2:

receive_dto.bullets1.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],0,-5))
                elif event.key == pygame.K_s:
                    if player_id == 0 and len(receive_dto.bullets0) <
```

```
2:

receive_dto.bullets0.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],0,5 ))
                    if player_id == 1 and len(receive_dto.bullets1) <
2:

receive_dto.bullets1.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],0,5))
                elif event.key == pygame.K_a:
                    if player_id == 0 and len(receive_dto.bullets0) <
2:

receive_dto.bullets0.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],-5,0 ))
                    if player_id == 1 and len(receive_dto.bullets1) <
2:

receive_dto.bullets1.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],-5,0))
                elif event.key == pygame.K_d:
                    if player_id == 0 and len(receive_dto.bullets0) <
2:

receive_dto.bullets0.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],5,0 ))
                    if player_id == 1 and len(receive_dto.bullets1) <
2:

receive_dto.bullets1.append(Bullet(player_id,receive_dto.player_x[play
er_id],receive_dto.player_y[player_id],5,0))
                elif event.key == pygame.K_q:
                    pygame.quit()
                    quit()

    if
view_rect_player.colliderect(pygame.Rect(receive_dto.player_x[opponent
_id],receive_dto.player_y[opponent_id], player_height, player_width)):
        players[opponent_id].draw(win)
    pygame.display.update()
    receive_dto.player_x[0] = players[0].x
    receive_dto.player_y[0] = players[0].y
    receive_dto.player_x[1] = players[1].x
    receive_dto.player_y[1] = players[1].y

    try:
        client.sendall(pickle.dumps(receive_dto))
        receive_dto = pickle.loads(client.recv(data_size))
    except Exception as e:
        run = False
```

```
        print("Couldn't get game")
        print("An error occurred:", e)
        break

    update_player_bullets(receive_dto)

    pygame.display.set_caption(
        f'(Red):{receive_dto.points[player_id]},'
        f'(Green):{receive_dto.points[opponent_id]}')
```