server.py

```
import socket
from _thread import *
import pickle
from components import *
from constants import ADDRESS, PORT, MAX_CLIENTS

# initiliaze the server socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.bind((ADDRESS, PORT))
except socket.error as e:
    str(e)

s.listen(MAX_CLIENTS)
print("Waiting for a connection, Server Started")

games = {} # list of Game instances
idCount = 0 # num of lcinets connected

# Start a thread for a client
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.reset()
                    elif data != "get":
                        game.shoot(p, data)

                    conn.send(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)
        games[gameId].players[p] = Player()
        games[gameId].current_player_id = p
        print("Creating a new game...")
    else:
        p = 1
        games[gameId].players[p] = Player()
        games[gameId].ready = True


    start_new_thread(threaded_client, (conn, p, gameId))
```

client.py

```
import pygame
from network import Network
from constants import *
from components import *
from utils import *
from math import floor
import time

# Initialize Pygame
pygame.init()
win = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("Client")

# Redraw Game Window
def redrawWindow(win, game, p):
    win.fill((0,0,0))

    if not(game.connected()):
        animation_time = int(time.time() * 2)
        dots_count = (animation_time % 4)
        dots = '.' * dots_count
```

```
        font = pygame.font.SysFont("assets/font.ttf", 28)
        text = font.render("WAITING FOR PLAYER" + dots, 1, (255,255,255), True)
        win.blit(text, (WINDOW_WIDTH/2 - text.get_width()/2, WINDOW_HEIGHT/2 -
text.get_height()/2))
    else:
        win.blit(radar_map, (12,54))
        win.blit(ocean_map, (12*2 + 342,54))

        draw_grid(win, game.players[p].grid)

        font = pygame.font.SysFont("assets/font.ttf", 20)

        if game.current_player_id == p:
            info_text = f"YOUR TURN. YOU HAVE {NUM_SHOOTS - game.shoot_count} SHOOTS!"
        else:
            info_text = f"PREPARE FOR THE OPPONENT'S SHOOTS"

        text = font.render(info_text, 1, (255, 255,255))
        text_rect = text.get_rect()
        text_rect.center = (WINDOW_WIDTH // 2, WINDOW_HEIGHT // 12)
        win.blit(text, text_rect)

        round_text = font.render(f"ROUND {(game.battle_count) // 2}", 1,
(255,255,255))
        round_text_rect = round_text.get_rect()
        round_text_rect.center = (WINDOW_WIDTH // 2, WINDOW_HEIGHT // 20)
        win.blit(round_text, round_text_rect)


        draw_ocean_mask(win, game.players[p].mask)
        draw_radar_mask(win, game.players[1 - p].mask)

    pygame.display.update()

# Load the Main Game Screen
def main():
    run = True
    clock = pygame.time.Clock()
    n = Network()
    player = int(n.getP())
    print("You are player", player)

    while run:
        clock.tick(60)
        try:
            game = n.send("get")
        except:
            run = False
            print("Couldn't get game")
            break

        if game.connected() and game.winner() != -1:
```

```
            redrawWindow(win, game, player)
            pygame.time.delay(500)

            font = pygame.font.SysFont("assets/font.ttf", 70)
            if (game.winner() == 1 and player == 1) or (game.winner() == 0 and player
== 0):
                text = font.render("YOU WON!", 1, (255, 255, 255))
            else:
                text = font.render("YOU LOST...", 1, (255, 255, 255))

            text_rect = text.get_rect()
            text_rect.center = (WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2)
            pygame.draw.rect(win, (0,0,0), (0, WINDOW_HEIGHT // 2 - 40, WINDOW_WIDTH,
80))

            win.blit(text, text_rect)
            pygame.display.update()

            try:
                game = n.send("reset")
            except:
                run = False
                print("Couldn't get game")
                break
            pygame.time.delay(3000)

        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()
                if RADAR_GRID_X < pos[0] < RADAR_GRID_X + GRID_WIDTH - TILE_SIZE and \
                    OCEAN_GRID_Y < pos[1] < OCEAN_GRID_Y + GRID_HEIGHT - TILE_SIZE:
                    pos_grid = [floor(abs(pos[i] - RADAR_GRID_POS[i])//(TILE_SIZE -
1)) for i in range(2)]
                    data = f"{pos_grid[0]},{pos_grid[1]}"
                    if game.current_player_id == player and game.connected():
                        n.send(data)

        redrawWindow(win, game, player)


# Load the Menu Screen
def menu_screen():
    run = True
    clock = pygame.time.Clock()

    blinking_counter = 0
```

```
    while run:
        clock.tick(60)
        win.fill((0,0,50))
        win.blit(titleScreen, (0,0))

        font = pygame.font.Font("assets/font.ttf", 18)
        text = font.render("PRESS SPACE TO PLAY", 1, (255,255,255))
        text_rect = text.get_rect()
        text_rect.center = (WINDOW_WIDTH // 2, 3 * (WINDOW_HEIGHT) // 4)

        if blinking_counter % 60 < 20:
            win.blit(text, text_rect)

        blinking_counter += 1
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                run = False
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    gameSounds['play'].set_volume(0.05)
                    gameSounds['play'].play()
                    run = False

    main()

if __name__ == '__main__':
    while True:
        menu_screen()
```

network.py

```
import socket
import pickle
from constants import ADDRESS, PORT

class Network:
    """Class that represents a client conection with the server
    """

    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.addr = (ADDRESS, PORT)
        self.p = self.connect()

    def getP(self):
        """Get the player id

        Returns:
```

```
            int: Player id
        """
        return self.p

    def connect(self):
        """Connect to the server

        Returns:
            int: Player id of the client sent by server
        """
        try:
            self.client.connect(self.addr)
            return self.client.recv(2048).decode()
        except:
            pass

    def send(self, data):
        """Send data to the server and receive the response

        Args:
            data (str): Client status

        Returns:
            Game : The Game instance sent by the server
        """
        try:
            self.client.send(str.encode(data))

            return pickle.loads(self.client.recv(2048*3))
        except socket.error as e:
            print(e)
```