МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Проект

З дисципліни «Комп’ютерні мережі»

**Виконав:** *студент 3-го курсу*

*гр. КА-71*

*Крохальов І.Д.*

**Прийняв:** *Кухарєв С.О*

**Київ 2020р.**

**Постановка задачі**

Мета:

Розробити продукт, що буде використовувати архітектуру “клієнт-сервер”, на прикладі відомої консольної гри “Танчики”.

Логіка гри:

Передбачено до 4 гравців.

Мета гравця — лишитися в живих та перемогти своїх суперників.

Початково у всіх health score 5, із кожним влучанням кулі, зменшується на 1. Якщо health score =0, гравець програв і завершив гру.

**Реалізація**

Засоби:

Надамо перелік технологій, що будуть використані для написання продукту.

Бібліотека pygame — для ігрової розробки в python 3.7.

Клієнт. Для реалізації клієнта використовуються модулі стандартної бібліотеки Python : socket, \_threads, pickle.

Сервер. Для сервера також знадобляться вказані модулі — socket, \_threads, pickle.

Перелік підзадач:

Розіб’ємо завдання на підзадачі, та до деяких надамо можливі особливостями реалізації.

1. Створити клас “Player”:

Навести опис класу Player, його змінних та методів(малювання, рух зображення).

1. Створити клас “Bullet”:

За ідеєю, куля — окремий від player клас. Він має власні змінні(розташування, колір, швидкість тощо).

1. Проміжна ланка між майбутніми клієнтом і сервером — class Network.

У Network є власні змінні(зберігає гравця та адресу), також власні методи, схожі на методи класу socket — send, receive.

1. Нарешті, перехід до основної частини. Реалізація файлу server.py

Сервер має підключати гравців до гри. Також контролює health score в процесі гри, якщо health score=0, то з’єднання переривається(GAME OVER).

1. Реалізація клієнта(файл client.py) — реалізація коду, потрібного для клієнта — малювання вікна, гравців, друк повідомлень про ваші влучення або про влучення суперника.

Опис класів та функцій:

Клієнт. У клієнті реалізовано багато функцій, таких як my\_target, enemy\_target (перевірка влучень), check\_bullet\_or\_player() - перевірка на межі вікна(щоб об’єкти завжди були у кадрі), redrawwindow...- для малювання вікна. Також експлуатується обьєкт класу Network.

Сервер. Сервер отримує дані про гравця та надсилає дані про трьох інших гравців. Також приймає та скасовує підключення.

Лістинг програми:

player.py:

import pygame

class Player():

def \_\_init\_\_(self, x, y, h1, h2 ,h3, health):

#координаты

self.x = x

self.y = y

self.health = health

#ширина-высота

self.width = 50

self.height = 50

self.color = (h1, h2, h3)

self.rect = (x,y,self.width,self.height)

#скорость передвижения за одно нажатие

self.vel = 2

def draw(self, win):

#отрисовка игрока

pygame.draw.rect(win, self.color, self.rect)

def move(self, width, height):

keys = pygame.key.get\_pressed()

#считали нажатую клавишу(вверх вниз вправо влево) и передвинули игрока)

if(( keys[pygame.K\_LEFT] ) and (self.x - self.vel > 0)):

self.x -= self.vel

if(( keys[pygame.K\_RIGHT]) and (self.x+self.vel < width)):

self.x += self.vel

if(( keys[pygame.K\_UP] ) and (self.y - self.vel > 0)):

self.y -= self.vel

if(( keys[pygame.K\_DOWN]) and (self.y + self.vel < height)):

self.y += self.vel

self.update()

bullet.py:

import pygame

class Bullet():

def \_\_init\_\_(self, x, y,direction ,color ):

# центр шарика

self.x = x

self.y = y

# радиус

self.r = 10

self.color = color

# направление, скорость

self.direction = direction

self.vel = 3 #1

def draw(self, win):

pygame.draw.circle(win, self.color, (self.x, self.y), self.r )

def move(self):

keys = pygame.key.get\_pressed()

# обработка нажатой клавиши

if self.direction == 1:

self.x -= self.vel

if self.direction == 3:

self.x += self.vel

if self.direction == 4:

self.y -= self.vel

if self.direction == 2:

self.y += self.vel

network.py:

import socket

import pickle #для работы с потоком байтов

class Network:

def \_\_init\_\_(self):

self.client = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_STREAM) #клиентский сокет

self.server = "192.168.1.140" #IP сервера

self.port = 5555 #порт сервера

self.addr = (self.server, self.port)

self.p = self.connect()

def get\_player(self):

return self.p

#выполнить соединение клиент-сервер, вернуть полученные данные с сокета, 2048 - размер буфера

def connect(self):

try:

self.client.connect(self.addr)

return pickle.loads(self.client.recv(2048))

except:

pass

def send(self, data):

try:

self.client.send(pickle.dumps(data))

return 0

except socket.error as e:

print(e)

def receive(self):

try:

return pickle.loads(self.client.recv(2048))

except socket.error as e:

print(e)

server.py:

import socket

from \_thread import \* #работа с потоками

from player import Player

from bullet import Bullet

import pickle

server = "192.168.1.140" #IP сервера

port = 5555 #порт сервера

s = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_STREAM) #серверный сокет

try:

s.bind((server, port))

except socket.error as e:

str(e)

s.listen(4) #прослушивание подключений . Максимум 4 игрока.

print("Waiting for a connection, Server Started")

players = [ Player(0, 0, 255, 0, 0, 5), \

Player(0, 450, 0, 255, 0, 5),\

Player(450, 0, 0, 0, 255, 5),\

Player( 450,450, 0, 200, 200, 5) ]

bullets = [None, None, None, None]

def threaded\_client(conn, player):

global players\_data

conn.send( pickle.dumps([player, players[player]]))

reply = " "

while True:

try:

# получили сообщение из conn

data = pickle.loads(conn.recv(2048))

players[data[0]] = data[1]

bullets[data[0]] = data[2]

# если сообщение пустое - напечатать Disconnected

if( not data ):

print("Disconnected")

break

else:

if player == 0:

#players[1] = data[0]

#players[2] = data[1]

#players[3] = data[2]

reply = [ [1,players[1],bullets[1]], [2,players[2], bullets[2]], [3, players[3], bullets[3]] ]

if player == 1:

#players[0] = data[0]

#players[2] = data[1]

#players[3] = data[2]

reply = [ [0,players[0],bullets[0]], [2,players[2], bullets[2]], [3, players[3], bullets[3]] ]

if player == 2:

#players[0] = data[0]

#players[1] = data[1]

#players[3] = data[2]

reply = [ [0,players[0],bullets[0]], [1,players[1], bullets[1]], [3, players[3], bullets[3]] ]

if player == 3:

#players[0] = data[0]

#players[1] = data[1]

#players[2] = data[2]

reply = [ [0,players[0],bullets[0]], [1,players[1], bullets[1]], [2, players[2], bullets[2]] ]

#print("Received: ", data)

#print("Sending : ", reply)

# Иначе отправить данные про трех остальных игроков.

conn.sendall(pickle.dumps(reply))

except:

break

# Если обмен сообщениями окончен - напечатать LOST CONNECTION

print("Lost connection")

conn.close()

current\_player = 0

while True:

conn, addr = s.accept()

print("Connected to:", addr)

start\_new\_thread(threaded\_client, (conn, current\_player) )

current\_player +=1

client.py:

import pygame

from network import Network

from player import Player

from bullet import Bullet

import sys

from \_thread import \*

import time

bullet = None

Number = None

q = True

player = None

metka = True

WHITE = (255, 255, 255)

BLACK = (0,0,0)

width = 500

height = 500

win = pygame.display.set\_mode((width, height))

pygame.display.set\_caption("Client")

def enemy\_target(player, bullets):

for bullet in bullets:

if( bullet != None):

#если пуля другого игрока есть и попала в Вас.

if( ( bullet.x < player.x+50 ) and (bullet.x > player.x) and (bullet.y < player.y+50 ) and(bullet.y > player.y) ):

player.health -=1

print('your health', player.health)

def my\_target(players, bullet):

for player in players:

# если вы выстрелили и попали в цель.

if( ( bullet.x < player.x+50 ) and (bullet.x > player.x) and (bullet.y < player.y+50 ) and(bullet.y > player.y) ):

print('nice shot')

print(len(players))

return True

return False

def move\_player():

global player

global width

global height

player.move(width-50,height-50)

#проверка на выход за пределы окна

def check\_bullet\_or\_player(bullet):

if( ( bullet.x > width ) or ( bullet.x < 0)):

return True

if( (bullet.y> height ) or ( bullet.y < 0) ):

return True

return False

def redrawWindow\_player( players, bullets ):

global player

global bullet

players.append(player)

bullets.append(bullet)

#win.fill(WHITE)

win.fill(BLACK)

for player in players:

player.draw(win)

for bullet in bullets:

if( bullet != None):

bullet.draw(win)

pygame.display.update()

players.pop(len(players)-1)

bullets.pop(len(bullets)-1)

def redrawWindow\_bullet(players, network):

global q

global bullet

global Number

global player

global metka

clock = pygame.time.Clock()

while True:

#win.fill(WHITE)

#redrawWindow\_player(players, [bullet])

#network.send([Number, players[0], bullet])

#bullet.draw(win)

if( bullet != None ):

bullet.move()

if( check\_bullet\_or\_player(bullet) or my\_target(players, bullet) ):

network.send([Number, player, bullet])

q = True

metka = True

bullet = None

return True

clock.tick(400)

else:

break

def main():

run = True

network = Network()

p = network.get\_player() #подключили игрока. теперь p - это тройка номер-игрок-пуля(None)

global player

player = p[1]

clock = pygame.time.Clock() #отсчет времени

global Number

Number = p[0]

global q

global bullet

global metka

while run:

clock.tick(60)

players = []

#закрытие окна

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.quit()

if( player.health < 1):

player.color = WHITE

redrawWindow\_player(players, bullets)

network.send([Number, player, bullet])

pygame.quit()

print( "You Lose")

break

keys = pygame.key.get\_pressed()

#обработка клавиш-выстрелов

if( q ):

if( keys[pygame.K\_a]):

bullet = Bullet(player.x+int(player.width/2), player.y+int(player.height/2), 1 ,player.color)

#n.send\_bullet(bullet, 1)

q = False

if( keys[pygame.K\_w]):

bullet = Bullet(player.x+int(player.width/2), player.y+int(player.height/2), 4 ,player.color)

#n.send\_bullet(bullet, 2)

q = False

if( keys[pygame.K\_d]):

q = False

bullet = Bullet(player.x+int(player.width/2), player.y+int(player.height/2), 3 ,player.color)

#n.send\_bullet(bullet, 3)

print('d')

if( keys[pygame.K\_s]):

q = False

bullet = Bullet(player.x+int(player.width/2), player.y+int(player.height/2), 2 ,player.color)

#n.send\_bullet(bullet, 4)

#if( bullet == None ):

network.send([Number, player, bullet])

data = network.receive()

players = [ data[0][1], data[1][1], data[2][1]]

bullets = [ data[0][2], data[1][2], data[2][2]]

enemy\_target(player, bullets)

#print( 'here')

if(( bullet != None) and (metka == True)):

start\_new\_thread( redrawWindow\_bullet, (players, network ) )

metka = False

move\_player()

redrawWindow\_player(players, bullets)

main()

Результати роботи:

Було створено спрощену багатокористувацьку гру “Танчики” із клієнт-серверною архітектурою.

Посилання на репозиторій:

<https://github.com/Krokha18/cn_simple_tanks>