24/04/23

TICTACTOE DUAL

Integrantes:
Pablo Armando Uscanga Camacho
Sebastián Astiazaran Lopez
Rolando Adrián Palacios Guzmán

Servidor en Lenguaje C:

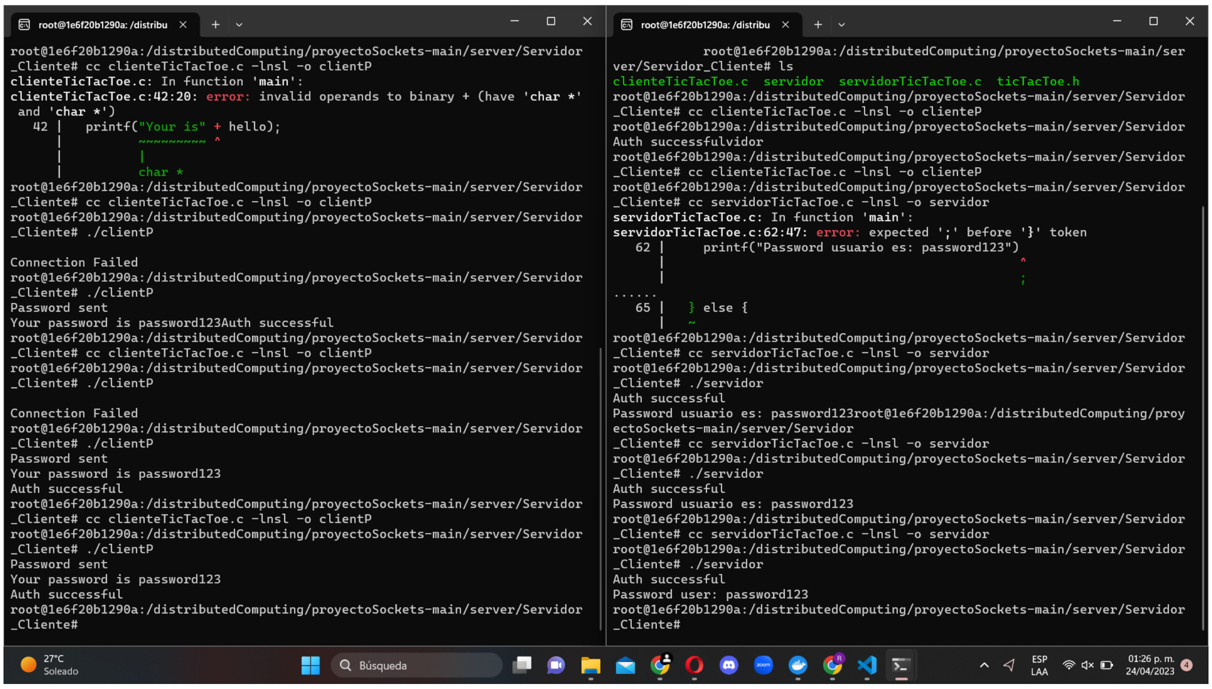
```c
//Servidor Autentificación

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

//#include "ticTacToe.h"

#define PORT 8080

int main(int argc, char const *argv[]) {
  int server_fd, new_socket, valread;
  struct sockaddr_in address;
  int opt = 1;
  int addrlen = sizeof(address);
  char buffer[1024] = {0};
  char *hello = "Auth successful";

  // Creating socket file descriptor
  if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("socket failed");
    exit(EXIT_FAILURE);
  }

  // Forcefully attaching socket to the port 8080
  if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt))) {
    perror("setsockopt");
    exit(EXIT_FAILURE);
  }
  address.sin_family = AF_INET;
  address.sin_addr.s_addr = INADDR_ANY;
  address.sin_port = htons(PORT);

  // Binding socket
  if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    perror("bind failed");
    exit(EXIT_FAILURE);
  }

  // Listening to incoming client requests
  if (listen(server_fd, 3) < 0) {
    perror("listen");
    exit(EXIT_FAILURE);
  }

  // Accepting incoming client requests
  if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen)) < 0) {
    perror("accept");

    exit(EXIT_FAILURE);
  }


  // Receiving client message and sending response
  valread = read(new_socket, buffer, 1024);
  if (strcmp(buffer, "password123") == 0) {
    send(new_socket, hello, strlen(hello), 0);
    printf("Auth successful\n");
   // ticTacToe();

  } else {
    printf("Auth failed\n");
  }
  return 0;

  if ( send(server_fd, hello, strlen(hello), 0) == -1) {
    perror("send");
    exit(1);
  }

  close(server_fd);
   printf("Conexion cerrada\n");
  return 0;

}
```

Cliente en C:

Aquí se muestra la variable hello que es igual a la contraseña: “Password123”

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define PORT 8080

int main(int argc, char const *argv[]) {
  int sock = 0, valread;
  struct sockaddr_in serv_addr;
  char *hello = "password123";
  char buffer[1024] = {0};

  // Creating socket
  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\n Socket creation error \n");
    return -1;
  }

  memset(&serv_addr, '0', sizeof(serv_addr));

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(PORT);

  // Convert IPv4 and IPv6 addresses from text to binary form
  if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
    printf("\nInvalid address/ Address not supported \n");
    return -1;
  }
```

```c
  // Connect to the server
  if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection Failed \n");
    return -1;
  }

  // Send authentication password to the server
  send(sock, hello, strlen(hello), 0);
  printf("Password sent\n");

  // Receive response from the server
  valread = read(sock, buffer, 1024);
  printf("%s\n", buffer);
  return 0;
}
```



```
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clientP
clienteTicTacToe.c: In function 'main':
clienteTicTacToe.c:42:20: error: invalid operands to binary + (have 'char *'
 and 'char *')
   42 |   printf("Your is" + hello);
      |          ~~~~~~~~~ ^
      |          |
      |          char *
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clientP
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./clientP

Connection Failed
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./clientP
Password sent
Your password is password123Auth successful
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clientP
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./clientP

Connection Failed
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./clientP
Password sent
Your password is password123
Auth successful
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clientP
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./clientP
Password sent
Your password is password123
Auth successful
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente#
```

```
            root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/ser
ver/Servidor_Cliente# ls
clienteTicTacToe.c  servidor  servidorTicTacToe.c  ticTacToe.h
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clienteP
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
Auth successfulvidor
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc clienteTicTacToe.c -lnsl -o clienteP
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc servidorTicTacToe.c -lnsl -o servidor
servidorTicTacToe.c: In function 'main':
servidorTicTacToe.c:62:47: error: expected ';' before '}' token
   62 |     printf("Password usuario es: password123")
      |                                               ^
      |                                               ;
......
   65 |   } else {
      |   ~
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc servidorTicTacToe.c -lnsl -o servidor
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./servidor
Auth successful
Password usuario es: password123root@1e6f20b1290a:/distributedComputing/proy
ectoSockets-main/server/Servidor
_Cliente# cc servidorTicTacToe.c -lnsl -o servidor
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./servidor
Auth successful
Password usuario es: password123
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# cc servidorTicTacToe.c -lnsl -o servidor
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente# ./servidor
Auth successful
Password user: password123
root@1e6f20b1290a:/distributedComputing/proyectoSockets-main/server/Servidor
_Cliente#
```

24/04/23

Codigo de TicTacToe.py:

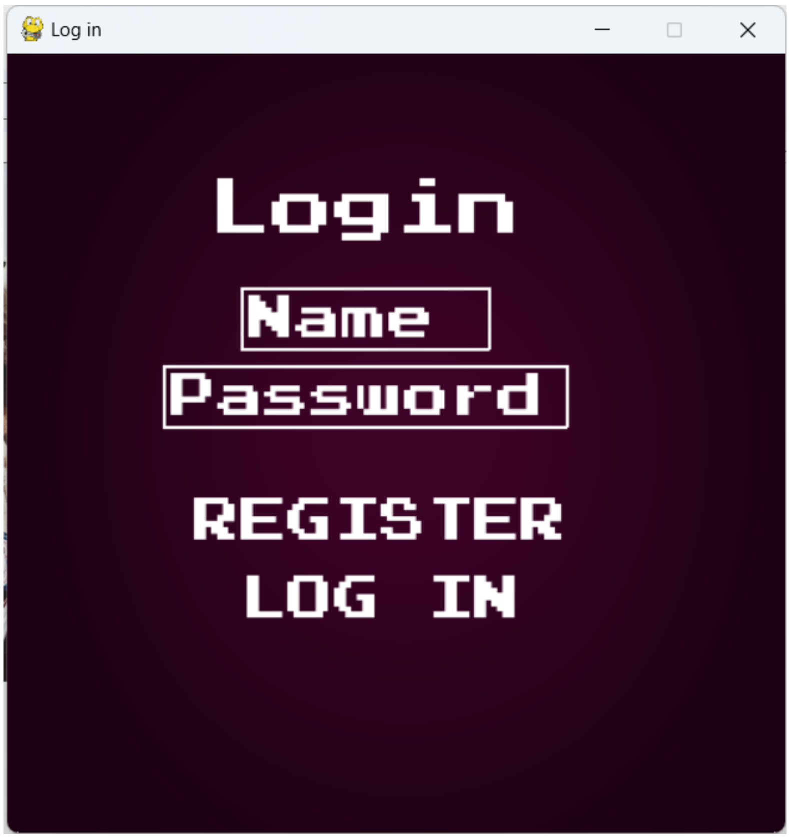
```
import pygame as pg
import sys
from random import randint
from button import Button
from login import InputBox
import time
import socket

"""
Proyecto sockets
Computo distribuido

Autores:
@Pablo Uscanga
@Sebastian Astiazaran
@Rolo boliche

"""

WINDOW_SIZE = 500
CELL_SIZE = WINDOW_SIZE // 3
BG = pg.image.load('./client/resources/Background.png')
INF = float('inf')
vec2 = pg.math.Vector2
CELL_CENTER = vec2(CELL_SIZE / 2)

stopTime = False
loginToken = False

###############################################################################
###################
###############################################################################
###################
###############################################################################
###################
###############################################################################
###################

class TicTacToe:
    def __init__(self, game):
        self.game = game
        self.fieldImage = self.getScaledImage(path =
'./client/resources/table.jpg', res = [WINDOW_SIZE] * 2)
        self.oImage = self.getScaledImage(path = './client/resources/o.png',
res = [CELL_SIZE] * 2)
        self.xImage = self.getScaledImage(path = './client/resources/x.png',
res = [CELL_SIZE] * 2)

        self.gameArray = [[INF, INF, INF],[INF, INF, INF],[INF, INF, INF]]
        self.player = randint(0, 1)

        self.lineIndicesArray = [[(0,0), (0,1), (0,2)],
                                    [(1,0), (1,1), (1,2)],
                                    [(2,0), (2,1), (2,2)],
                                    [(0,0), (1,0), (2,0)],
                                    [(0,1), (1,1), (2,1)],
                                    [(0,2), (1,2), (2,2)],
                                    [(0,0), (1,1), (2,2)],
                                    [(0,2), (1,1), (2,0)]]

        self.winner = None
        self.gameSteps = 0
        self.font = pg.font.SysFont('Verdana', CELL_SIZE // 4, True)

    def checkWinner(self):
        for line_indices in self.lineIndicesArray:
            sumLine = sum([self.gameArray[i][j] for i, j in line_indices])
            if sumLine in {0, 3}:
                self.winner = 'XO'[sumLine == 0]
                self.winnerLine = [vec2(line_indices[0][::-1]) * CELL_SIZE +
CELL_CENTER,
                                   vec2(line_indices[2][::-1]) * CELL_SIZE +
CELL_CENTER]

    def runGameProcesss(self):
        global stopTime
        if stopTime:
            currentCell = vec2(pg.mouse.get_pos()) // CELL_SIZE
            col, row = map(int, currentCell)
            leftClick = pg.mouse.get_pressed()[0]
            if leftClick and self.gameArray[row][col] == INF and not
self.winner:
                self.gameArray[row][col] = self.player
                self.player = not self.player
                self.gameSteps += 1
                self.checkWinner()

    def drawObjects(self):
        for y, row in enumerate(self.gameArray):
            for x, obj in enumerate(row):
                if obj != INF:
                    self.game.screen.blit(self.xImage if obj else self.oImage,
vec2(x, y) * CELL_SIZE)

    def drawWinner(self):
        if self.winner:
            pg.draw.line(self.game.screen, 'red', *self.winnerLine, CELL_SIZE
// 8)
            label0 = self.font.render(f'Player "{self.winner}" wins!', True,
'white', 'black')
            self.game.screen.blit(label0, (WINDOW_SIZE // 2 -
label0.get_width() // 2, WINDOW_SIZE // 4))


    def draw(self):
        self.game.screen.blit(self.fieldImage, (0,0))
        self.drawObjects()
        self.drawWinner()

    @staticmethod
    def getScaledImage(path, res):
        img = pg.image.load(path)
        return pg.transform.scale(img, res)

    def printCaption(self):
        pg.display.set_caption(f'Player "{"OX" [self.player]}" turn')
        if self.winner:
            pg.display.set_caption(f'Player " {self.winner}" wins! Press space
to restart or enter to go back')
        elif self.gameSteps == 9:
            pg.display.set_caption(f'Game Tied! Press space to restart or
enter to go back')
```

```python
    def run(self):
        self.printCaption()
        self.draw()
        self.runGameProcesss()

##############################################################################
###################
##############################################################################
###################
##############################################################################
###################
##############################################################################
###################

class Game:

    def __init__(self):
        pg.init()
        self.screen = pg.display.set_mode([WINDOW_SIZE] * 2)
        self.clock = pg.time.Clock()
        self.tictactoe = TicTacToe(self)

    def newGame(self):
        self.tictactoe = TicTacToe(self)

    def checkEvents(self):
        global stopTime
        for event in pg.event.get():
            if event.type == pg.QUIT:
                pg.quit()
                sys.exit()
            if event.type == pg.KEYDOWN:
                if event.key == pg.K_SPACE:
                    self.newGame()
                if event.key == pg.K_RETURN:
                    stopTime = False
                    self.newGame()
                    self.manager()

    def get_font(size): # Returns Press-Start-2P in the desired size
        return pg.font.Font("./client/resources/font.ttf", size)
    def howToPlay(self):
        while True:
            OPTIONS_MOUSE_POS = pg.mouse.get_pos()

            game.screen.fill("white")

            OPTIONS_TEXT0 = Game.get_font(30).render("How to play", True,
"Black")
            OPTIONS_TEXT1 = Game.get_font(15).render("Just press the play
button", True, "Black")
            OPTIONS_TEXT2 = Game.get_font(15).render("each player will have
one turn", True, "Black")
            OPTIONS_TEXT3 = Game.get_font(15).render("whoever gets 3 in line",
True, "Black")
            OPTIONS_TEXT4 = Game.get_font(15).render("WINS!", True, "Black")
            OPTIONS_RECT0 = OPTIONS_TEXT0.get_rect(center=(250, 80))
            OPTIONS_RECT1 = OPTIONS_TEXT1.get_rect(center=(250, 120))
            OPTIONS_RECT2 = OPTIONS_TEXT2.get_rect(center=(250, 140))
            OPTIONS_RECT3 = OPTIONS_TEXT3.get_rect(center=(250, 160))
            OPTIONS_RECT4 = OPTIONS_TEXT4.get_rect(center=(250, 180))
            game.screen.blit(OPTIONS_TEXT0, OPTIONS_RECT0)
            game.screen.blit(OPTIONS_TEXT1, OPTIONS_RECT1)
            game.screen.blit(OPTIONS_TEXT2, OPTIONS_RECT2)
            game.screen.blit(OPTIONS_TEXT3, OPTIONS_RECT3)
            game.screen.blit(OPTIONS_TEXT4, OPTIONS_RECT4)


            OPTIONS_BACK = Button(image=None, pos=(250, 250),
                                text_input="BACK", font=Game.get_font(30),
base_color="Black", hovering_color="Green")

            OPTIONS_BACK.changeColor(OPTIONS_MOUSE_POS)
            OPTIONS_BACK.update(game.screen)

            for event in pg.event.get():
                if event.type == pg.QUIT:
                    pg.quit()
                    sys.exit()
                if event.type == pg.MOUSEBUTTONDOWN:
                    if OPTIONS_BACK.checkForInput(OPTIONS_MOUSE_POS):
                        Game.manager(self)
            pg.display.update()

    def registerUser(self):
        global loginToken
        pg.display.set_caption("Register User")
        registerMousePos = pg.mouse.get_pos()

        registrationUsernameBox = InputBox(150, 150, 160, 40, "Name")
        registrationPasswordBox = InputBox(100, 200, 260, 40, "Password")

        registrationUsernameBox.draw(game.screen)
        registrationPasswordBox.draw(game.screen)
        pg.display.flip()

        while True:
            for event in pg.event.get():
                game.screen.blit(BG, (0,0))

                registerText = Game.get_font(40).render("Registration", True,
"white")
                registerRect = registerText.get_rect(center=(250, 100))
                game.screen.blit(registerText, registerRect)

                registrationUsernameBox.draw(game.screen)
                registrationPasswordBox.draw(game.screen)
                registrationUsernameBox.handle_event(event)
                registrationPasswordBox.handle_event(event)

                registerMousePos = pg.mouse.get_pos()
                registerSux = Button(image=None, pos=(220, 350),
                                    text_input="REGISTER",
font=Game.get_font(30), base_color="white", hovering_color="yellow")
                registerBack = Button(image=None, pos=(220, 400),
                                    text_input="BACK", font=Game.get_font(30),
base_color="white", hovering_color="yellow")

                registerBack.changeColor(registerMousePos)
                registerSux.changeColor(registerMousePos)

                registerSux.update(game.screen)
                registerBack.update(game.screen)
                if event.type == pg.QUIT:
                    pg.quit()
                    sys.exit()
                if event.type == pg.MOUSEBUTTONDOWN:
                    if registerBack.checkForInput(registerMousePos):
                        Game.manager(self)
                    if registerSux.checkForInput(registerMousePos):
                        Game.manager(self)
                        loginToken = True

            pg.display.update()

    def login(self):

        pg.display.set_caption("Log in")

        usernameBox = InputBox(150, 150, 160, 40, "Name")
        passwordBox = InputBox(100, 200, 260, 40, "Password")

        loginRun = True

        usernameBox.draw(game.screen)
        passwordBox.draw(game.screen)
        pg.display.flip()

        while loginRun:
            for event in pg.event.get():
                global loginToken
                loginToken = False
                game.screen.blit(BG, (0,0))
                usernameBox.draw(game.screen)
                passwordBox.draw(game.screen)
                usernameBox.handle_event(event)
                passwordBox.handle_event(event)

                registerMousePos = pg.mouse.get_pos()
                registerText = Game.get_font(40).render("Login", True,
"white")
                registerRect = registerText.get_rect(center=(230, 100))
                game.screen.blit(registerText, registerRect)
                registerButton = Button(image=None, pos=(240, 300),
                                    text_input="REGISTER",
font=Game.get_font(30), base_color="white", hovering_color="yellow")
                loginButton = Button(image=None, pos=(240, 350),
                                    text_input="LOG IN",
font=Game.get_font(30), base_color="white", hovering_color="yellow")

                registerButton.changeColor(registerMousePos)
                registerButton.update(game.screen)

                loginButton.changeColor(registerMousePos)
                loginButton.update(game.screen)

                pg.display.flip()

                if event.type == pg.QUIT:
                    loginRun = False
                    pg.quit()
                if event.type == pg.MOUSEBUTTONDOWN:
                    if registerButton.checkForInput(registerMousePos):
                        Game.registerUser(self)
                    if loginButton.checkForInput(registerMousePos):
                        loginToken = True
                        Game.manager(self)
            pg.display.update()

    def manager(self):
        while True:

            #Inicio login
            ##############################
            global loginToken
            game.screen.blit(BG, (0,0))
            if loginToken == False:
                Game.login(self)

            ##############################
            game.screen.blit(BG, (0, 0))

            MENU_MOUSE_POS = pg.mouse.get_pos()

            MENU_TEXT = Game.get_font(50).render("MAIN MENU", True, "#b68f40")
            MENU_RECT = MENU_TEXT.get_rect(center=(250, 100))

            PLAY_BUTTON =
Button(image=pg.image.load("./client/resources/PlayRect.png"), pos=(250, 170),
                                text_input="PLAY", font=Game.get_font(40),
base_color="#d7fcd4", hovering_color="White")
            OPTIONS_BUTTON =
Button(image=pg.image.load("./client/resources/OptionsRect.png"), pos=(250,
240),
                                text_input="HOW TO PLAY",
font=Game.get_font(40), base_color="#d7fcd4", hovering_color="White")
            QUIT_BUTTON =
Button(image=pg.image.load("./client/resources/QuitRect.png"), pos=(250, 310),
                                text_input="QUIT", font=Game.get_font(40),
base_color="#d7fcd4", hovering_color="White")

            game.screen.blit(MENU_TEXT, MENU_RECT)

            for button in [PLAY_BUTTON, OPTIONS_BUTTON, QUIT_BUTTON]:
                button.changeColor(MENU_MOUSE_POS)
                button.update(game.screen)

            for event in pg.event.get():
                if event.type == pg.QUIT:
                    pg.quit()
                    sys.exit()
                if event.type == pg.MOUSEBUTTONDOWN:
                    if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                        while True:
                            global stopTime
                            game.screen.fill("white")
                            self.tictactoe.run()
                            if stopTime == False:
                                time.sleep(1)
                                stopTime = True
                            self.checkEvents()
                            pg.display.update()
                            self.clock.tick(60)
                    if OPTIONS_BUTTON.checkForInput(MENU_MOUSE_POS):
                        Game.howToPlay(self)
                    if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                        pg.quit()
                        sys.exit()

            pg.display.update()

##############################################################################
###################
##############################################################################
###################
##############################################################################
###################
##############################################################################
###################

if __name__ == '__main__':
    game = Game()
    game.manager()
```

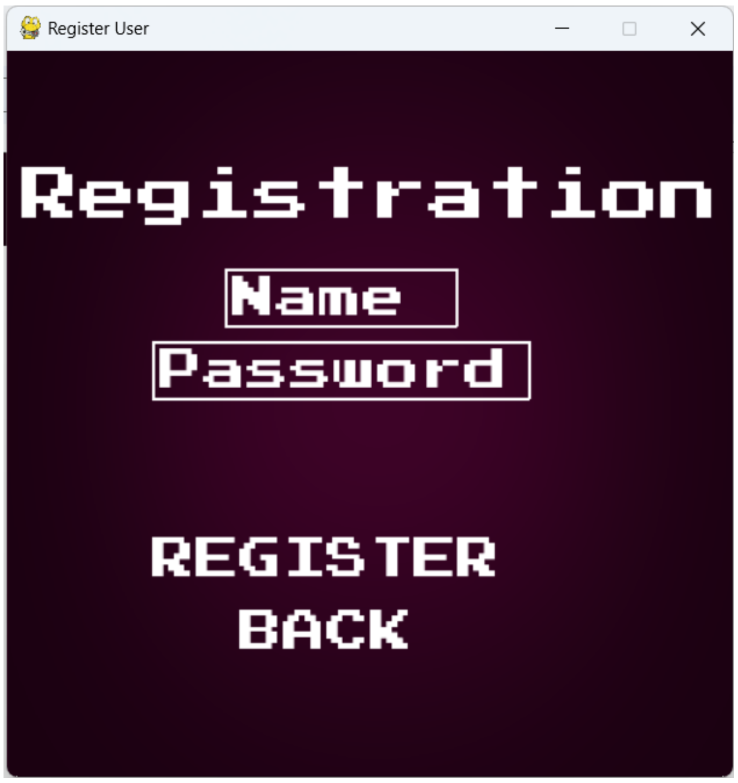
Video explicativo:
https://github.com/Sastiazaran/proyectoSockets/blob/main/explicacionInterfaz.mkv

Interfaz de juego:

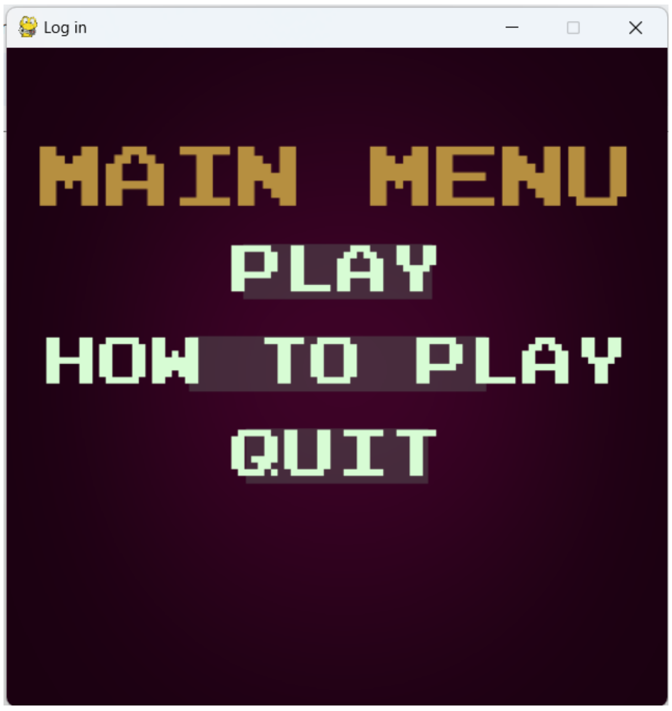
- Login:



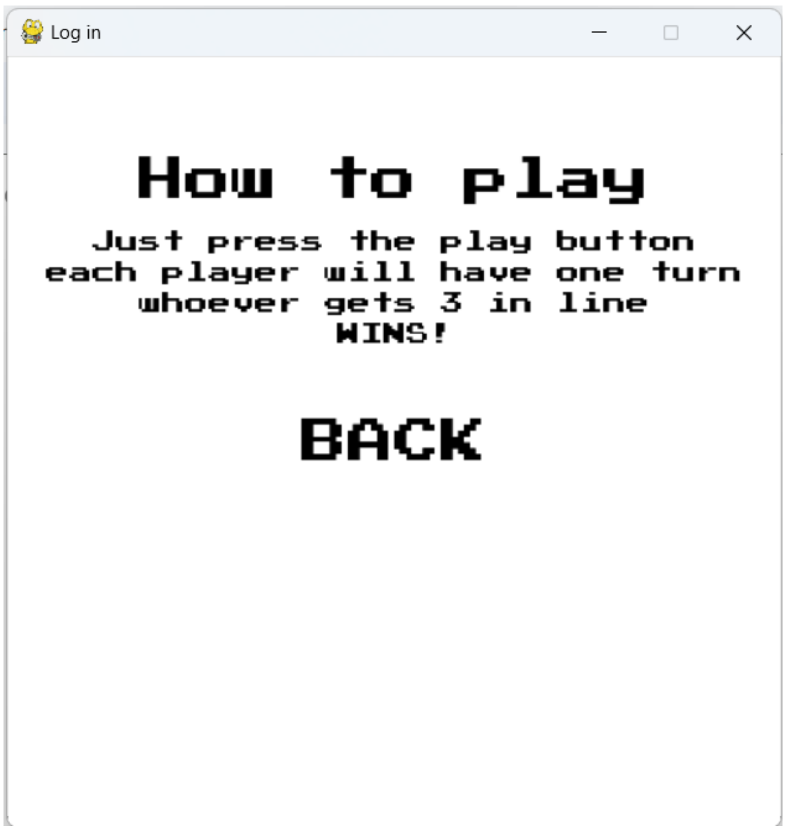
- Registro:



Pantalla de Carga:



Instrucciones:



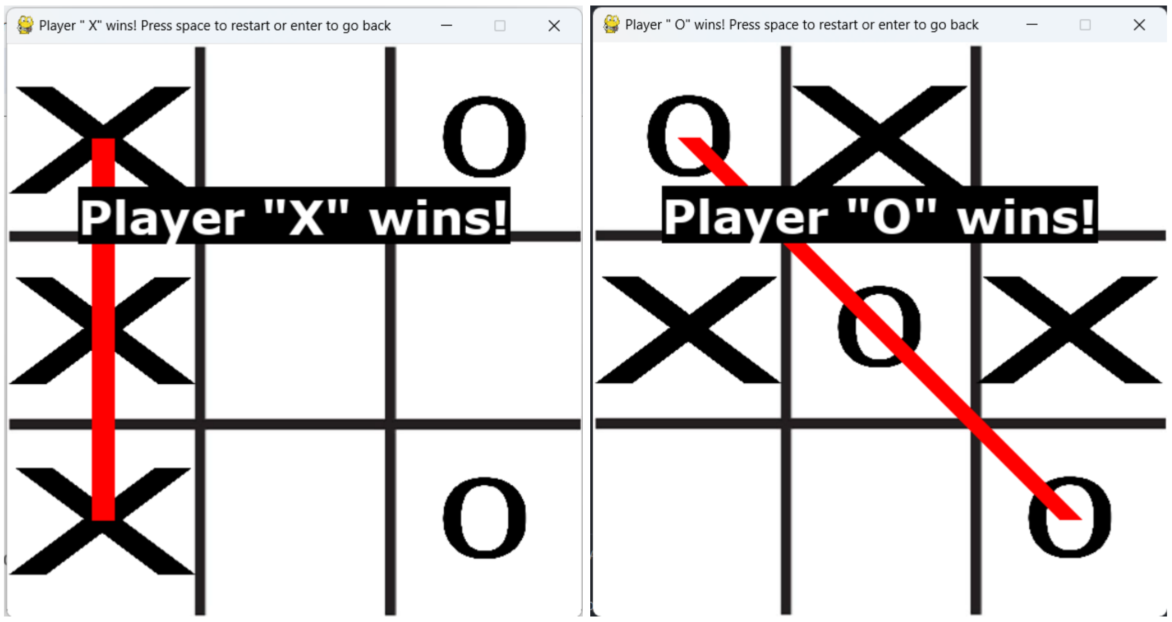
Log in

How to play

Just press the play button
each player will have one turn
whoever gets 3 in line
WINS!

BACK

Pantallas de Carga GANADORA:



Player " X" wins! Press space to restart or enter to go back

Player "X" wins!

Player " O" wins! Press space to restart or enter to go back

Player "O" wins!