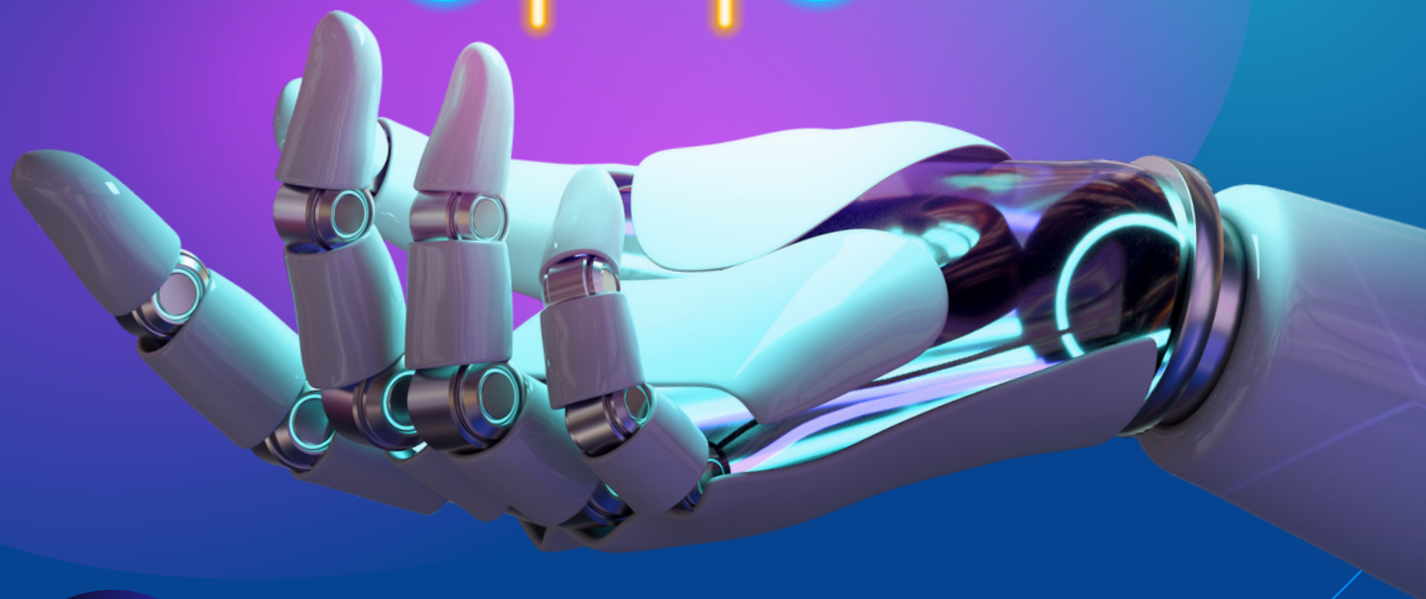
MANUAL TECNICO

TIC TAC TOE CON IA



Python

CARÁTULA

INTELIGENCIA ARTIFICIAL
ING. STEVEEN BLANCO
PROYECTO DE CURSO

Desarrolladores

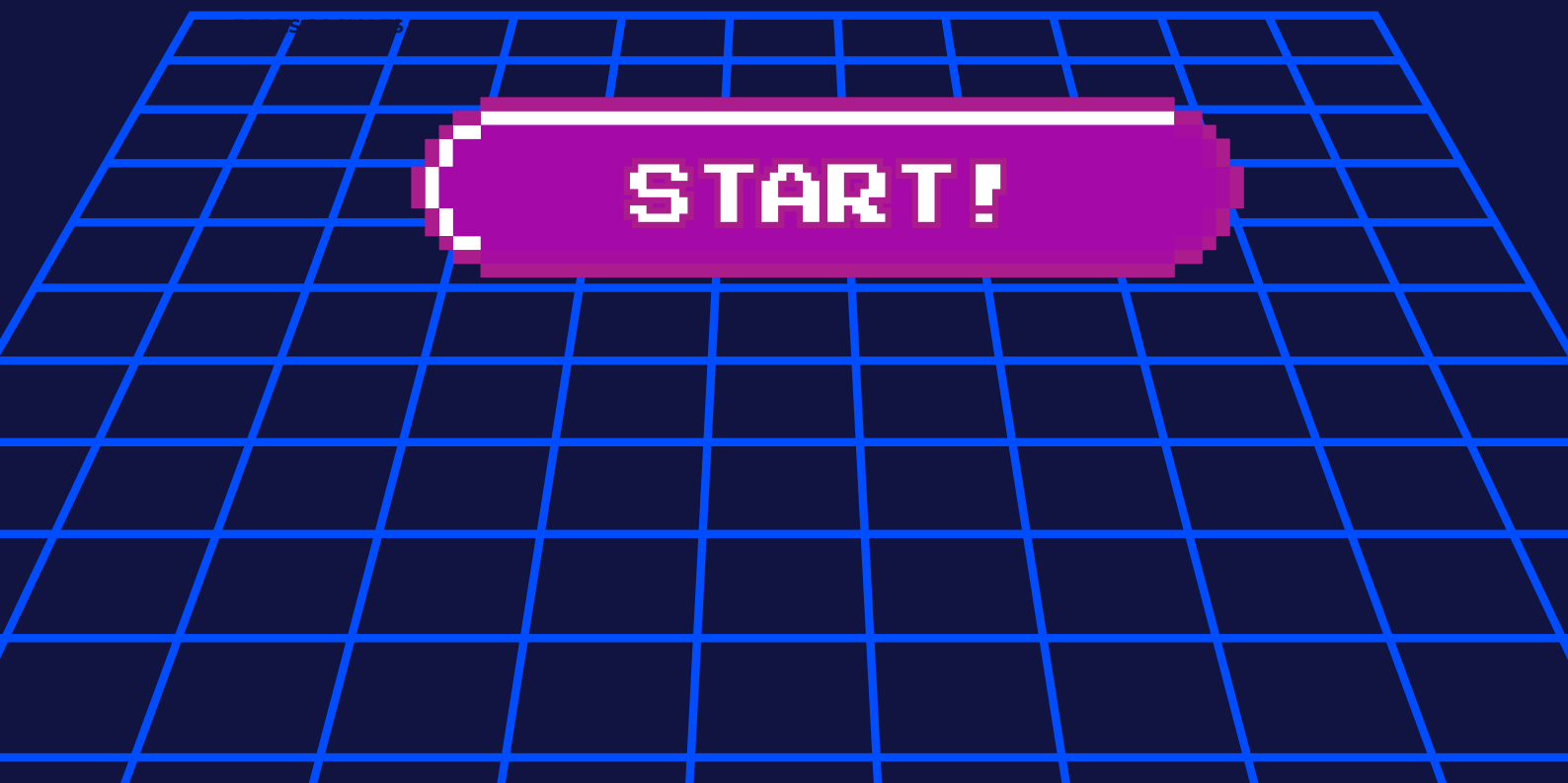
David Joaquin Ramirez Muñoz
1904002003

Edvin Stuardo Pérez Garcia
1904002011

INTRODUCCIÓN

Bienvenido al manual técnico del videojuego Tic Tac Toe con IA desarrollado en Python. A través de este documento exploraremos los aspectos técnicos y las funcionalidades implementadas en este proyecto.

Se considera que la aplicación de IA en los videojuegos es un área con un crecimiento significativo dentro de la industrial digital.



ASPECTOS TÉCNICOS

Herramienta

Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Visual Studio Code

ASPECTOS TÉCNICOS

Lenguaje

Python

Python es un lenguaje de programación de alto nivel, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas.

ASPECTOS TÉCNICOS

Bibliotecas

Pygame

Pygame es una biblioteca de Python que permite crear videojuegos y aplicaciones multimedia de manera sencilla. Funciona proporcionando herramientas para gestionar gráficos, sonidos, colisiones y eventos del usuario en una ventana de visualización. Esto permite a los desarrolladores crear juegos interactivos y animaciones utilizando Python de manera eficiente.

| pip Install pygame

ASPECTOS TÉCNICOS

NumPy

NumPy es una biblioteca de Python utilizada para realizar cálculos numéricos eficientes, especialmente en arreglos multidimensionales. Funciona proporcionando una estructura de datos llamada "array" que permite realizar operaciones matemáticas rápidas y optimizadas en grandes conjuntos de datos, utilizando funciones vectorizadas en lugar de bucles explícitos. Esto aumenta significativamente la velocidad de procesamiento en comparación con las operaciones estándar.

| pip install numpy

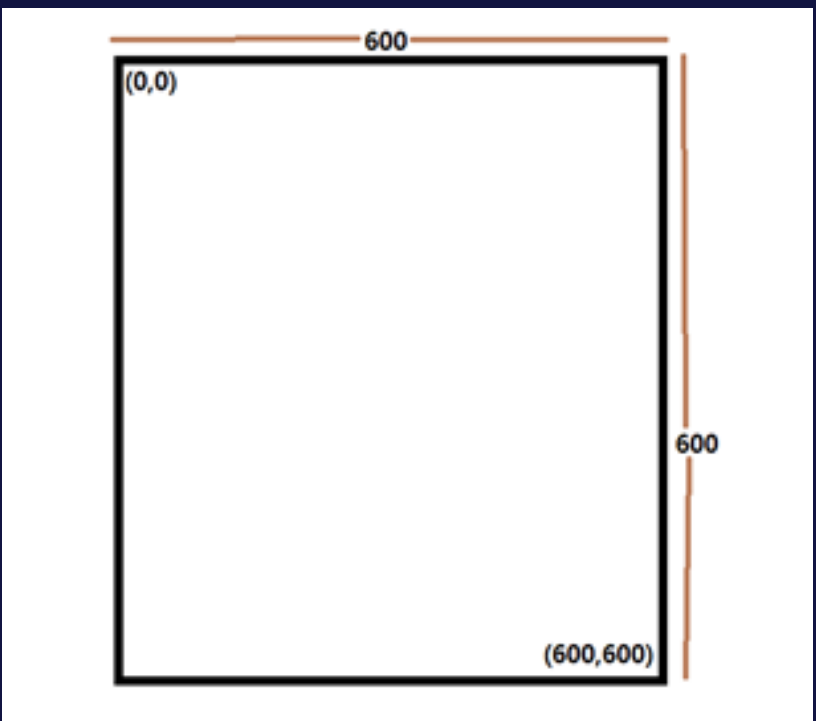
ASPECTOS TÉCNICOS

Elementos

Board

En Python, "Board" se refiere generalmente a una estructura de datos que representa un tablero de juego, como un tablero de ajedrez o un juego de mesa. Se utiliza para almacenar el estado actual del juego y facilitar las operaciones de manipulación y visualización dentro del mismo.

Diagrama del proyecto:
En la siguiente ilustración se muestra el diseño del tablero que se utilizó para la visualización de pygame:



ASPECTOS TÉCNICOS

Clases

Board

Esta clase contiene la integración del tablero para el videojuego, además se encuentran definidos los siguientes métodos:

__init__(self): Consiste en el método constructor de la clase Board, inicia el tablero en donde cada casilla está vacía, las cuales representan a los ceros dentro de una matriz, así también inicia los números de casillas.

final_state(self, show=False): Verifica si existe algún ganador del juego, en caso de que show muestre "True", entonces se hará el dibujo de una línea sobre aquellas casillas ganadoras, esto dentro de la interfaz gráfica: Retorno

- 0 = No hay jugador.
- 1 = Jugador 1 gana.
- 2 = Jugador 2 gana.

ASPECTOS TÉCNICOS

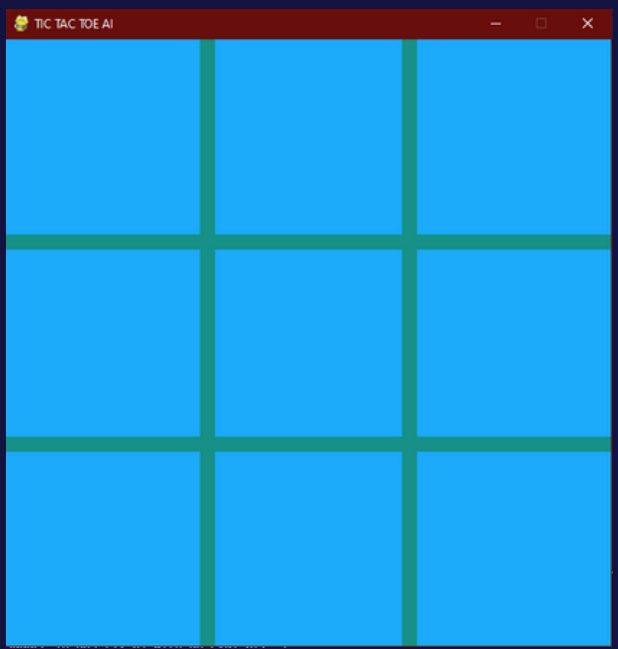
mark_sqr(self, row, col, player): Marca el signo respectivo en la casilla para cada jugador (1 y 2).

empty_sqr(self, row, col): Verifica si existe alguna casilla vacía.

get_empty_sqrs(self): Se encarga de retornar una lista de tuplas que expresan las coordenadas de las casillas que se encuentran vacías dentro del tablero.

isfull(self): Evalúa si cada casilla del tablero esté marcada.

isempty(self): Este hace la tarea contraria al método anterior, verifica si cada casilla del tablero está vacía.



ASPECTOS TÉCNICOS

AI

Esta clase contiene la integración de la inteligencia artificial para el videojuego, además se encuentran definidos los siguientes métodos:

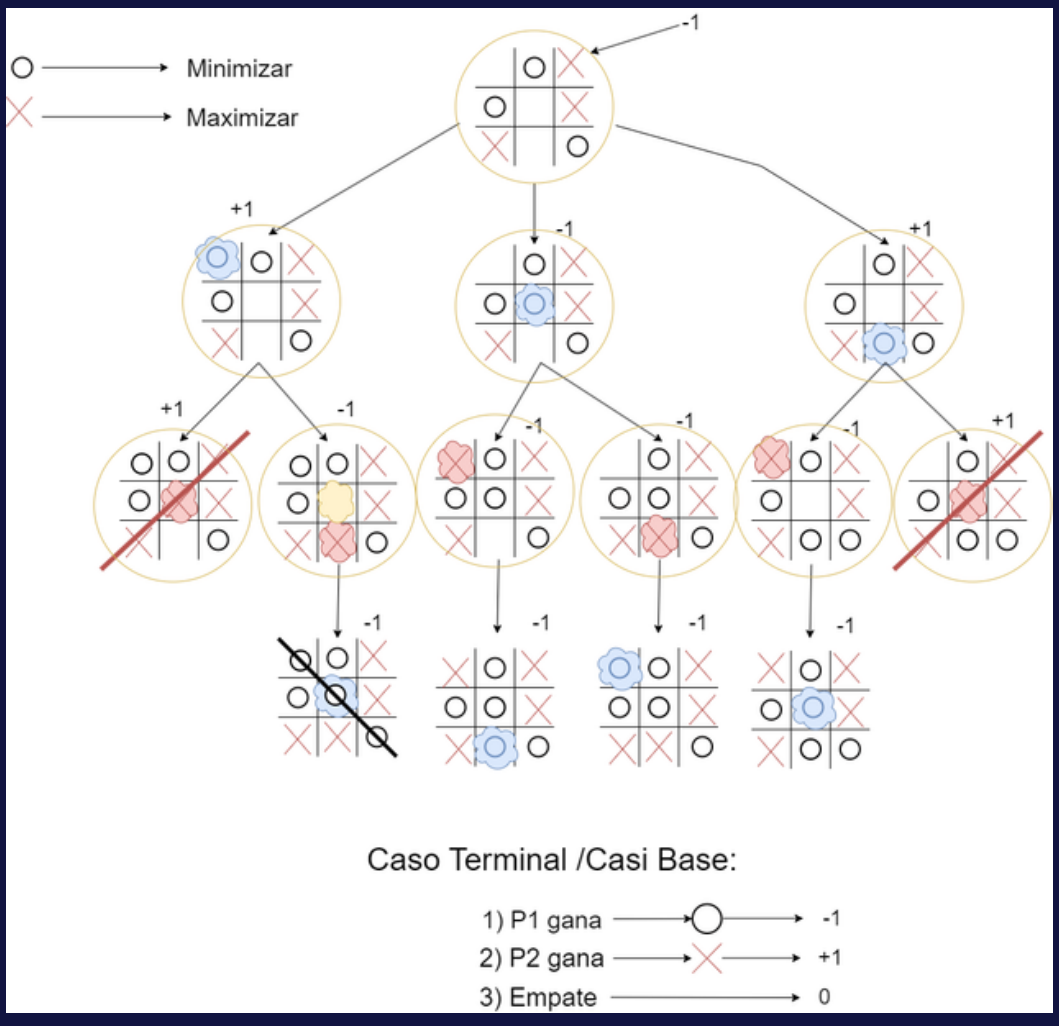
__init__(self, level=1, player=2): Consiste en el método constructor, inicia la inteligencia artificial con cierto nivel de dificultad y el jugador 2 que representa a la IA.

rnd(self, board): Expresa una estrategia random. Escoge una casilla vacía random y retorna sus coordenadas.

minimax(self, board, maximizing): Integra el algoritmo minimax con el fin de tomar decisiones en función al árbol de juego. Calcula una potencial mejor jugada para el jugador self.player y retorna el valor de evaluación de la jugada, así mismo las coordenadas de la casilla que representa la mejor jugada.

ASPECTOS TÉCNICOS

eval(self, main_board): A partir de este método, la IA decide su jugada. En caso la dificultad es igual a 0, entonces la IA decide jugar random, en caso contrario, utiliza el algoritmo minimax para encontrar la mejor jugada. Por último, se muestra la decisión qu tomó la IA y retorna las coordenadas de la casilla selccionada.



ASPECTOS TÉCNICOS

Game

Esta clase se encarga de manejar la lógica del videojuego, así también la interfaz gráfica. Además se encuentran definidos los siguientes métodos:

__init__(self): Inicia el juego, crea 2 objetos, uno de la clase board para representar el tablero y otro de la clase IA para representar la IA. Inicia al jugador 1 (X), la modalidad del juego (VS la IA) y llama a ShowLines para dibujar las líneas divisoras del tablero.

show_lines(self): Por medio de Pygame, dibuja las líneas divisoras del tablero.

draw_fig(self, row, col): En base a la posición indicada en el tablero, dibuja las figuras "X"/"O"

make_move(self, row, col): Con el jugador actual, marca la casilla y dibuja la figura en el tablero, luego da paso al siguiente jugador dandole el turno.

ASPECTOS TÉCNICOS

next_turn(self): Cambio el turno al otro jugador.

change_gamemode(self): Se encarga de cambiar la modalidad del videojuego, es decir:

- PVP: Jugador vs Jugador (Humanos)
- IA: Jugador vs IA (Humano vs Máquina)

isover(self): Verifica si ya ha terminado el juego, ya sea en caso de que un jugador ganó o bien, el tablero está lleno.

reset(self): Reinicio de juego.

ASPECTOS TÉCNICOS

Códigos

Altura y anchura de pygame

```
#---- ALTURA PYGAME-----
WIDTH = 600
HEIGHT = 600
```

Color de fondo

```
#----COLOR DEL FONDO----
BG_COLOR = (28,170,250)
```

ASPECTOS TÉCNICOS

Colores para líneas, bordes, radios de "O" y "X", así como espacios de borde.

```
#----OTRO COLOR-----
RED = (255,0,0)
LINE_COLOR = (23,145,135)
LINE_WIDTH = 15
BOARD_ROWS = 3
BOARD_COLS = 3
SQUARE_SIZE = WIDTH//BOARD_COLS
CIRCLE_RADIUS = SQUARE_SIZE//3
CIRCLE_WIDTH = 15
CIRCLE_COLOR = (239, 231, 200)
CROSS_WIDTH = 25
CROSS_COLOR = (66, 66, 66)
SPACE = SQUARE_SIZE//4
```

Color y nombre de la ventana de Pygame

```
screen = pygame.display.set_mode((WIDTH,HEIGHT))
pygame.display.set_caption('Tic Tac Toe AI')
screen.fill(BG_COLOR)
```

ASPECTOS TÉCNICOS

Tablero al hacer clic en cada cuadro con "O" y "X" correspondientes.

```
#board
board = np.zeros((BOARD_ROWS, BOARD_COLS))
#print(board)
```

Dibujar líneas para dividir el tablero 3x3

```
#---Dibujar Lineas---
def draw_line():
    # 1 Horizontal
    pygame.draw.line(screen, LINE_COLOR,(0, SQUARE_SIZE),(WIDTH,SQUARE_SIZE),LINE_WIDTH)
    # 2 Horizontal
    pygame.draw.line(screen, LINE_COLOR,(0,2 * SQUARE_SIZE),(WIDTH,2 * SQUARE_SIZE),LINE_WIDTH)
    # 1 Vertical
    pygame.draw.line(screen, LINE_COLOR,(SQUARE_SIZE,0),(SQUARE_SIZE,HEIGHT),LINE_WIDTH)
    # 2 Vertical
    pygame.draw.line(screen, LINE_COLOR,(2 * SQUARE_SIZE,0),(2 * SQUARE_SIZE,HEIGHT),LINE_WIDTH)
```

Dibujar líneas "O" y "X" en cada cuadro del tablero "Board" según lo mencionado anteriormente.

```
def draw_figures():
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            if board[row][col] == 1:
                pygame.draw.circle(screen, CIRCLE_COLOR, (int(col * SQUARE_SIZE + SQUARE_SIZE//2), int(row *
            elif board[row][col] == 2:
                pygame.draw.line(screen, CROSS_COLOR, (col * SQUARE_SIZE + SPACE, row * SQUARE_SIZE + SQUARE
                pygame.draw.line(screen, CROSS_COLOR, (col * SQUARE_SIZE + SPACE, row * SQUARE_SIZE + SPACE)
```

ASPECTOS TÉCNICOS

Identificar disponibilidad o contenido ("X" o "O") en cuadro del tablero "Board".

```
#-Esta disponible el cuadro del medio-
def mark_square(row, col, player):
    board[row][col] = player

def available_square(row, col):
    return board[row][col] == 0
def is_board_full():
    for row in range(BOARD_ROWS):
     for col in range(BOARD_COLS):
      if board[row][col] == 0:
        return False
    return True
```

ASPECTOS TÉCNICOS

Identificar si los cuadros del tablero "Board" contienen "X" u "O" para determinar la victoria del jugador en forma horizontal o vertical.

```python
#--Ganardor de lista----
def check_win(player):
    # vertical win check
    for col in range(BOARD_COLS):
        if board[0][col] == player and board[1][col] == player and board[2][col] == player:
            draw_vertical_winning_line(col, player)
            return True

    #horizontal win check
    for row in range(BOARD_ROWS):
        if board[row][0] == player and board[row][1] == player and board[row][2] == player:
            draw_horizontal_winning_line(row, player)
            return True

    #asc diagonal win check
    if board[2][0] == player and board[1][1] == player and board[0][2] == player:
        draw_asc_diagonal(player)
        return True

    #desc diagonal win check
    if board[0][0] == player and board[1][1] == player and board[2][2] == player:
        draw_desc_diagonal(player)
        return True

    return False
```

ASPECTOS TÉCNICOS

```python
def draw_vertical_winning_line(col, player):
    posX = col * SQUARE_SIZE + SQUARE_SIZE//2

    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR

    pygame.draw.line(screen, color, (posX, 15), (posX, HEIGHT - 15), 15)

def draw_horizontal_winning_line(row, player):
    posY = row * SQUARE_SIZE + SQUARE_SIZE//2

    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR

    pygame.draw.line(screen, color, (15, posY), (WIDTH - 15, posY), 15)

def draw_asc_diagonal(player):
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR

    pygame.draw.line(screen, color, (15, HEIGHT - 15), (WIDTH - 15, 15), 15)
```

```python
def draw_desc_diagonal(player):
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR

    pygame.draw.line(screen, color, (15, 15), (WIDTH - 15, HEIGHT - 15), 15)
```

ASPECTOS TÉCNICOS

Reinicio de Juego

```
#---Reiniciar partida----
def restart():
    screen.fill(BG_COLOR)
    draw_line()
    player = 1
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            board[row][col] = 0
```

Identificación de los dibujos de "X" y "O", además también incluye al jugador e identificación cuando se finalizó la partida

```
draw_line()
player = 1
game_over = False
```

ASPECTOS TÉCNICOS

Inicio y cierre del juego, incluyendo el inicio de la partida y la ventana del juego con Pygame

```
# ----MainLoop----
while True:
    for event in pygame.event.get():
     if event.type == pygame.QUIT:
        sys.exit()

     if event.type == pygame.MOUSEBUTTONDOWN and not game_over:
        mouseX = event.pos[0] #x
        mouseY = event.pos[1] #y

        clicked_row = int(mouseY // SQUARE_SIZE)
        clicked_col = int(mouseX // SQUARE_SIZE)

        if available_square(clicked_row, clicked_col):
            mark_square(clicked_row, clicked_col, player)
            if check_win(player):
                game_over = True
            player = player % 2 + 1

            draw_figures()
     if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_r:
            restart()
            game_over = False

    pygame.display.update()
```

ASPECTOS TÉCNICOS

Alojamiento

Github

Es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones llamado Git.

ASPECTOS TÉCNICOS

Github

Desarrollo del Juego: Se realizó un total de 13 commits, elaborados por los desarrolladores, dando así a la creación del videojuego.

Intrucciones Juegos:

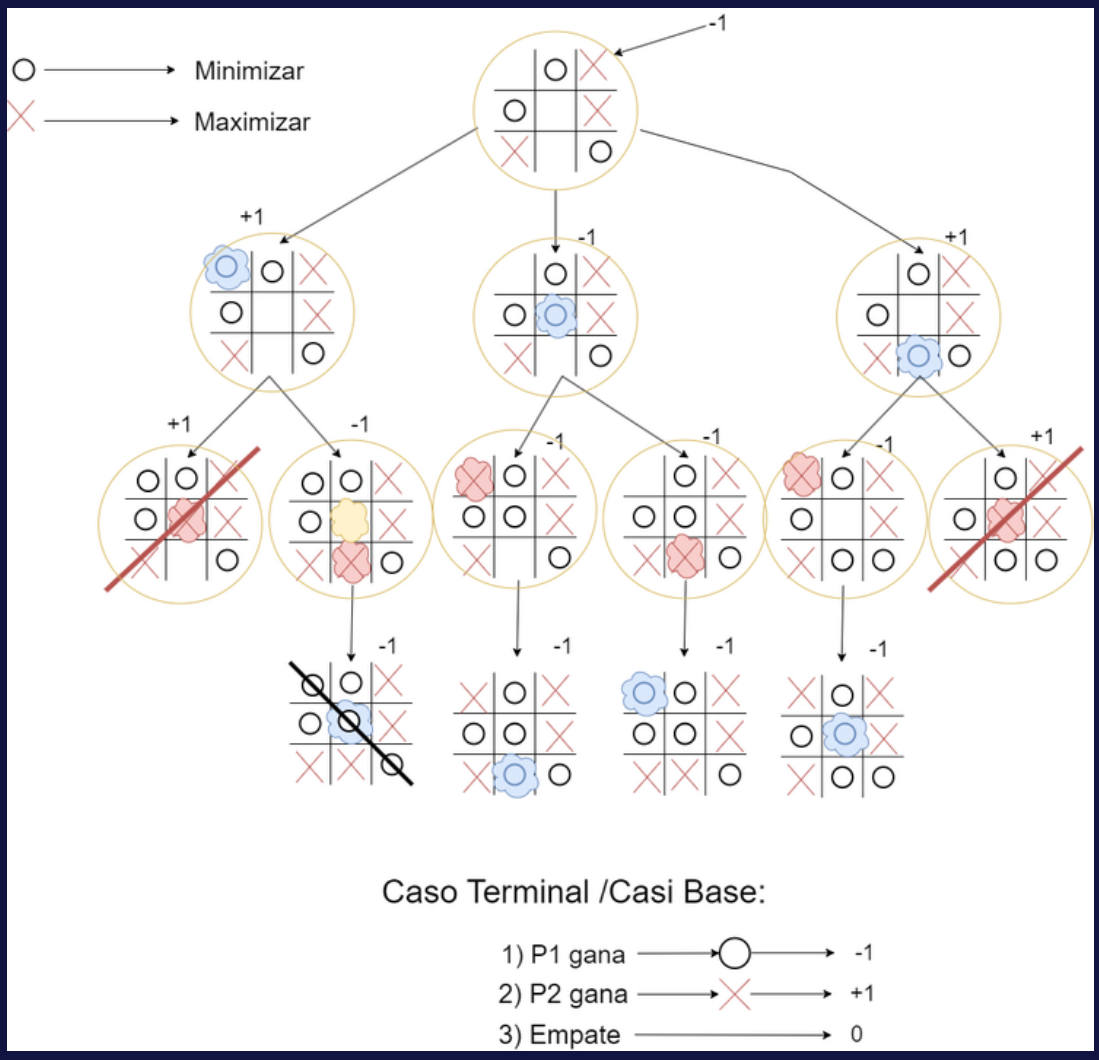
```
- presiona 'g' para cambiar el modo de juego (pvp o ai)
- presione '0' para cambiar el nivel de IA a 0 (aleatorio)
- presione '1' para cambiar el nivel de IA a 1 (imposible)
- presiona 'r' para reiniciar el juego
```

Link a repositorio en Github

ASPECTOS TÉCNICOS

Algoritmo minimax

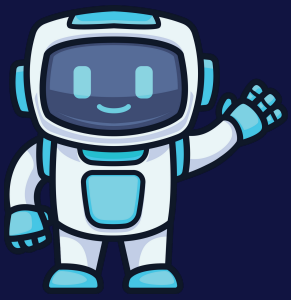
Es un método de decisión para minimizar la pérdida máxima esperada en juegos con adversario y con información perfecta. El objetivo del jugador puede resumirse en: elegir el mejor movimiento para ti mismo suponiendo que tu contrincante escogerá el peor movimiento para ti



Algoritmo Minimax del juego - IA

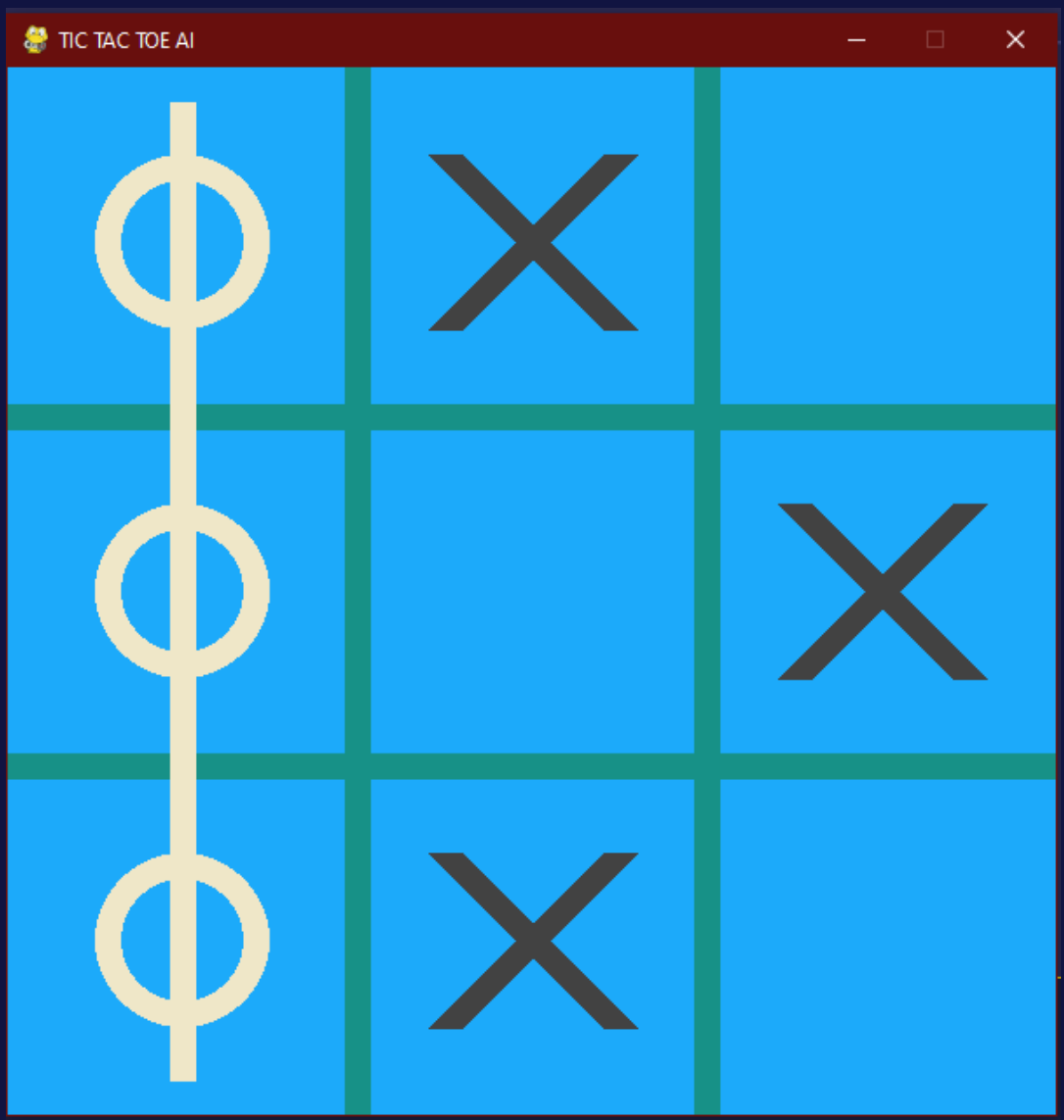
ASPECTOS TÉCNICOS

Resultado final



X: Jugador Humano

Y: Inteligencia Artificial




FINISH
RESTART