UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

CENTRO UNIVERSITARIO EL NARANJO, MIXCO

FACULTAD DE INGENIERÍA EN SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN

ING. JOSE MIGUEL VILLATORO HIDALGO.

CURSO PROGRAMACION III.

TIC TAC TOE.

MANUAL TECNICO,

MANUAL DE USUARIO

ELABORADO POR

CARNET	NOMBRE	Sección	PORCENTAJE
9490-22-1157	Josué Sebastián Mancilla González	A	100%
9490-22-958	Anthony Fabian Ramírez Orellana	A	100%
9490-22-4974	Oscar José Cojulún Mendoza	A	100%
9490-22-3432	Willy Estuardo Culajay Asturias	A	100%

Mayo 2024

Manual técnico

Este manual describe técnicamente el código de nuestro juego Tic Tac Toe, utilizando Pygames para la interfaz gráfica y Graphviz para visualizar el árbol de decisiones de las jugadas, el programa esta echo para que el humano juegue contra la máquina, donde la maquina puede jugar más o menos difícil aprendiendo del cómo se nos desempeña el juego.

Requisitos del sistema:

- Tener instalado Python 3.6 o superior.
- Instalar Pygame, de ser necesario instalarlo con “pip install pygame”.
- Instalar Graphviz, de ser necesario instalarlo con “pip install graphviz”.

Estructura del código: El código cuenta con constantes de configuración para definir aspectos visuales del juego, como dimensiones de la ventana, colores y grosores de las líneas, algunas de las cuales son:

```python
# Configuración de la pantalla
ANCHO = 600
ALTO = 600

FILAS = 3
COLUMNAS = 3
TAM_CASILLA = ANCHO // COLUMNAS

GROSOR_LINEA = 15
GROSOR_CIRCULO = 15
GROSOR_CRUZ = 20

RADIO = TAM_CASILLA // 4

OFFSET = 50
COLOR_FONDO = (0, 150, 100)
COLOR_LINEA = (50, 50, 50)
COLOR_CIRCULO = (50, 50, 50)
COLOR_CRUZ = (250, 230, 250)
```

Clases:

“Tablero”: Se encarga de gestionar el estado del tablero del juego e incluye el marcado de casillas, verifica si una casilla está vacía y determina el estado final del juego.

Descripción de los métodos/funciones:

```python
class Tablero:
    def __init__(self):
        self.casillas = np.zeros((FILAS, COLUMNAS))
        self.casillas_vacias = self.casillas
        self.casillas_marcadas = 0

    def estado_final(self, mostrar=False):
        '''
        @devuelve 0 si es empate
        @devuelve 1 si jugador 1 es el ganador
        @devuelve 2 si jugador 2 es el ganador
        '''

        # Victoria vertical
        for col in range(COLUMNAS):
            if self.casillas[0][col] == self.casillas[1][col] == self.casillas[2][col] != 0:
                if mostrar:
                    color = COLOR_CIRCULO if self.casillas[0][col] == 2 else COLOR_CRUZ
                    pos_inicial = (col * TAM_CASILLA + TAM_CASILLA // 2, 20)
                    pos_final = (col * TAM_CASILLA + TAM_CASILLA // 2, ALTO - 20)
                    pygame.draw.line(pantalla, color, pos_inicial, pos_final, GROSOR_LINEA)
                return self.casillas[0][col]
```

“__init__:”: inicializa el tablero con una matriz de ceros y establece el conteo de casillas marcadas.

“estado_final(self, mostrar=False):”: Determina el estado final del juego (victoria, empate o en progreso). Si “mostrar” es True, dibuja las lineas de victoria en la pantalla.

“marcar_casilla(self, fila, col, jugador):”: Marca una casilla en el tablero con el número del jugador.

“casilla_vacia(self, fila, col):”: Verifica si una casilla especifica está vacía.

“obtener_casillas_vacias(self):”: Devuelve una lista de todas las casillas vacías en el tablero.

“esta_lleno(self):”: Verifica si el tablero está lleno.

“esta_vacio(self):”: Verifica si el tablero este vacío.

“Maquina”: Define la lógica de la máquina, que juega dependiendo de nuestras partidas jugadas.

```
111  class Maquina:
112      def __init__(self, nivel=1, jugador=2):
113          self.nivel = nivel
114          self.jugador = jugador
115
116      def aleatorio(self, tablero):
117          casillas_vacias = tablero.obtener_casillas_vacias()
118          idx = random.randrange(0, len(casillas_vacias))
119          return casillas_vacias[idx]
120
121      def minimax(self, tablero, maximizando):
122          estado = tablero.estado_final()
123
```

Descripción de métodos:

“__init__(self, nivel=1, jugador=2):”: Inicializa la maquina con un nivel de dificultad y el número de jugador.

“aleatorio(self, tablero):”: Selecciona un movimiento aleatorio entre las casillas vacías del tablero.

“minimax(self, tablero, maximizando):”: Implementa el algoritmo minimax para evaluar el mejor movimiento posible.

“evaluar(self, tablero_principal):”: Evalúa el movimiento basado en el nivel de dificultad de la máquina.

“Juego”: Gestiona la lógica del juego, incluyendo la interacción con Pygame para la interfaz gráfica, la actualización del estado del juego y la visualizacion del árbol de decisiones usando Graphviz.

```
171  class Juego:
172      def __init__(self):
173          self.tablero = Tablero()
174          self.maquina = Maquina()
175          self.jugador = 1
176          self.modo_juego = 'maquina'
177          self.en_juego = True
178          self.mostrar_lineas()
179          self.arbol = Digraph(comment='Jugadas de Tic Tac Toe')
180          self.arbol.graph_attr['ranksep'] = '1.5'
181          self.arbol.graph_attr['nodesep'] = '0.5'
182          self.arbol.graph_attr['splines'] = 'true'
183          self.estado_actual = str(self.tablero.casillas)
184          self.arbol.node(self.estado_actual, self.estado_actual)
```

Descripción de métodos:

“__init__(self):”: Inicializa el juego, creando el tablero, la maquina y configurando los estados iniciales del juego.

“mostrar_lineas(self):”: Dibuja las líneas del tablero en la pantalla.

“dibujar_figura(self, fila, col):”: Dibuja la figura correspondiente (X o O) en la casilla seleccionada.

“hacer_movimiento(self, fila, col):”: Realiza un movimiento en el tablero y actualiza el estado del juego.

“siguiente_turno(self):”: Alterna el turno del jugador.

“cambiar_modo_juego(self):”: cambia el modo de juego entre humano y máquina.

“terminado(self):”: Verifica si el juego ha terminado.

“reiniciar(self):”: Reinicia el juego.

“actualizar_arbol(self, fila, col):”: Actualiza el arbol de decisiones con el movimiento actual.

Funciones principales:

Inicialización y configuración: La inicialización de Pygame y la configuración del tablero se manejan al comienzo del programa.

```
28  # Inicializar Pygame
29  pygame.init()
30  pantalla = pygame.display.set_mode((ANCHO, ALTO))
31  pygame.display.set_caption('Juego de Tic Tac Toe')
32  pantalla.fill(COLOR_FONDO)
```

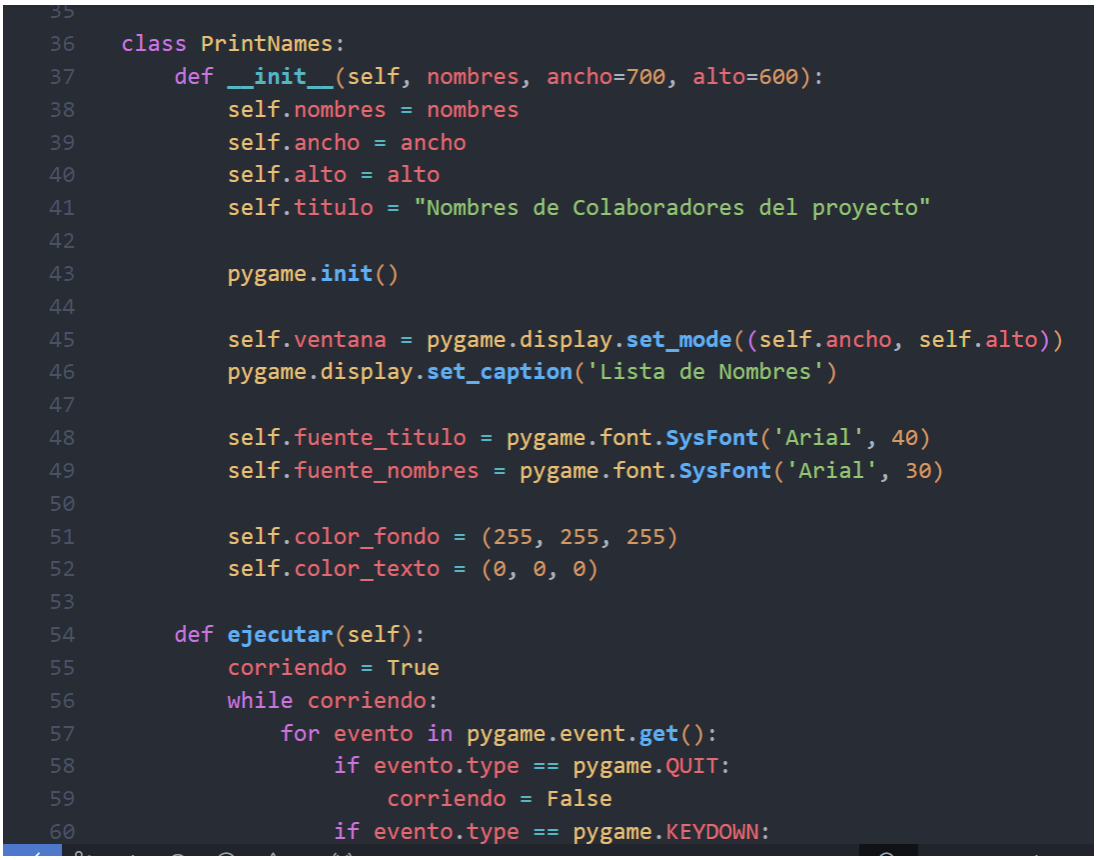
Eventos y lógica del juego: El bucle principal del juego maneja eventos como los clics del ratón y las teclas presionadas, y actualiza el estado del juego en consecuencia.

```
242 def main():
243     juego = Juego()
244     tablero = juego.tablero
245     maquina = juego.maquina
246 
247     while True:
248         for evento in pygame.event.get():
249             if evento.type == pygame.QUIT:
250                 pygame.quit()
251                 sys.exit()
252 
253             if evento.type == pygame.KEYDOWN:
254                 if evento.key == pygame.K_g:
255                     juego.cambiar_modo_juego()
```

"printnames": Esta clase esta diseñada para mostrar una lista de nombres en una ventana grafica utilizando la librería Pygame, útil para presentar los nombres de los integrantes de nuestro grupo.

Descripción y métodos:

La clase se encarga de inicializar y manejar la ventana y renderizar los nombres en pantalla.



```
class PrintNames:
    def __init__(self, nombres, ancho=700, alto=600):
        self.nombres = nombres
        self.ancho = ancho
        self.alto = alto
        self.titulo = "Nombres de Colaboradores del proyecto"

        pygame.init()

        self.ventana = pygame.display.set_mode((self.ancho, self.alto))
        pygame.display.set_caption('Lista de Nombres')

        self.fuente_titulo = pygame.font.SysFont('Arial', 40)
        self.fuente_nombres = pygame.font.SysFont('Arial', 30)

        self.color_fondo = (255, 255, 255)
        self.color_texto = (0, 0, 0)

    def ejecutar(self):
        corriendo = True
        while corriendo:
            for evento in pygame.event.get():
                if evento.type == pygame.QUIT:
                    corriendo = False
                if evento.type == pygame.KEYDOWN:
```

“__init__(self, nombres, ancho=700, alto=600)”: Es el constructor de la clase que inicializa los atributos y la configuración de la ventana.

Algunos de los parámetros son: nombres-lista de nombres a mostrar, ancho-es el ancho de la ventana por defecto de 700, alto-alto de la ventana por defecto de 600.

“ejecutar(self)”: método principal que ejecuta el bucle de eventos de Pygame y renderiza el titulo y los nombres en la ventana.

Este inicia un bucle que permanece activo mientras “corriendo” es “True”, mientras se manejan los eventos de cierre de la ventana y pulsación de la tecla “ECS”, llena la ventana con un color de fondo, renderiza y muestra el titulo centrado en la parte superior, renderiza y muestra cada nombre de la lista con un espacio vertical uniforme, actualiza la pantalla de Pygame para reflejar los cambios.

Librerías utilizadas:

1. Pygame: Es una librería de Python diseñada para escribir y desplegar interfaces entre otras funciones para videojuegos. Incluye funcionalidades para manejar gráficos, sonidos y eventos, lo que facilita la creación de juegos y aplicaciones multimedia.

En el proyecto se utiliza para dibujar el tablero de juego y las figuras (X y O), también maneja los eventos del usuario como los clics del ratón y las teclas presionadas, tambien actualiza la pantalla en tiempo real durante el juego.

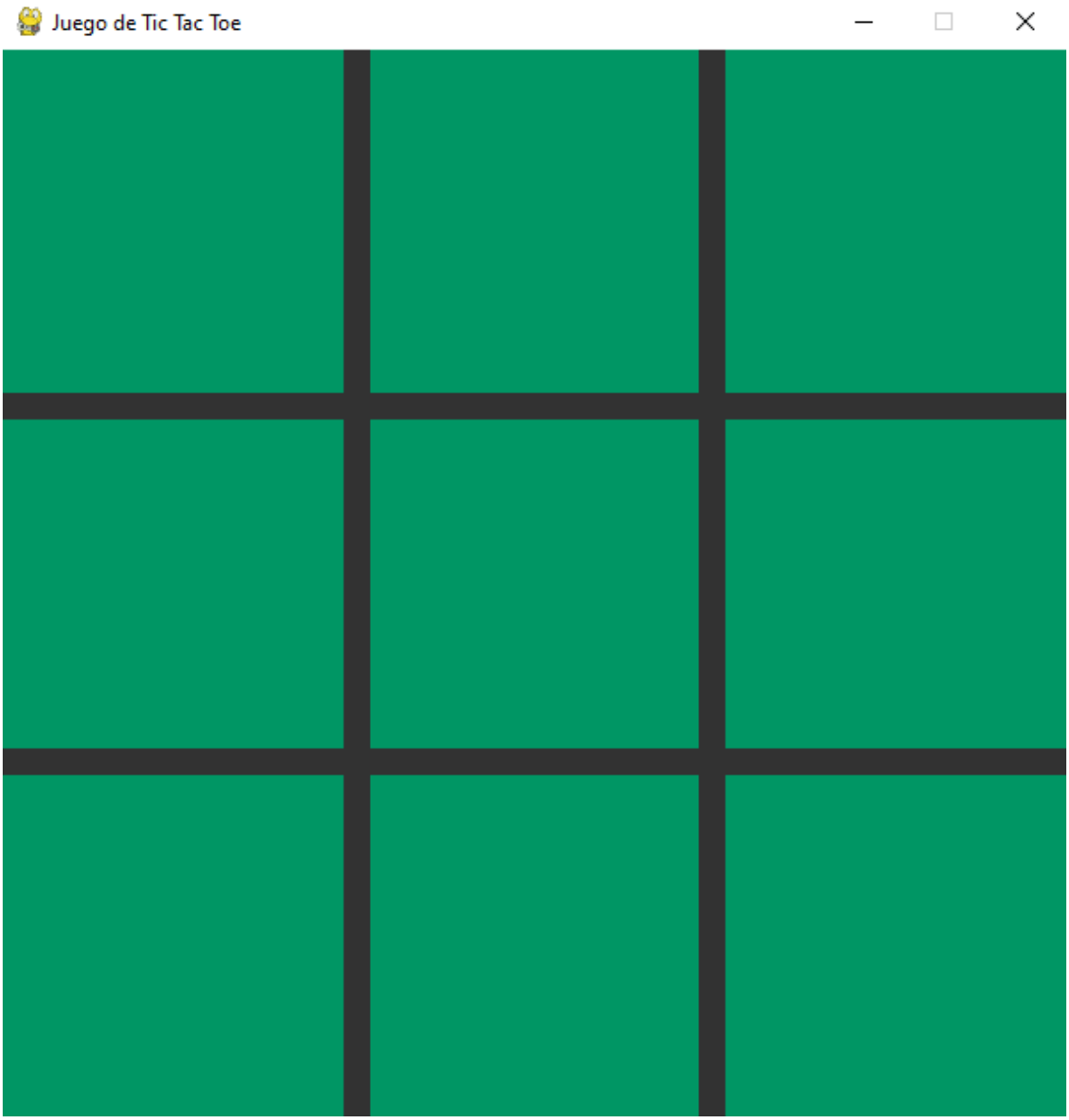
2. Graphviz: Es una librería de software para la presentación de gráficos, utiliza el lenguaje DOT para describir gráficos y genera visualizaciones en varios formatos.
   En el proyecto se utiliza para visualizar el árbol de decisiones de las jugadas, también guarda el árbol en formato SVG para mejor visualización.
3. Copy: Proporciona funciones para crear copias superficiales y profundas de objetos en Python, "copy.copy(obj)" por ejemplo, crea una copia superficial del objeto, mientras que "copy.deepcopy(obj)" crea una copia profunda, copiando recursivamente todos los objetos contenidos.
   En el programa se utiliza el "copy.deepcopy" para crear copias profundas del estado del tablero cuando se ejecuta el algoritmo minimax. Esto permite evaluar posibles movimientos sin alterar el estado actual del juego.
4. Sys: Proporciona acceso a algunas variables y funciones que interactúan fuertemente con el intérprete de Python, por ejemplo: "sys.exit()" termina el programa, "sys.argv" lista de argumentos de línea de comandos pasados al script, "sys.stdout" flujo de salida estándar de consola.
   En el programa se utiliza el "sys.exit()" para salir del programa cuando se detecta un evento de cierre de ventada en Pygame.
5. Random: Proporciona funciones para generar números aleatorios y realiza selecciones aleatorias, por ejemplo: "random.randrange(start, stop)" devuelve un numero entero seleccionado aleatoriamente del rango "[start, stop)", "random.choice(seq)" devuelve un elemento seleccionado aleatoriamente de la secuencia "seq".
   En el progmra se utiliza para seleccionar aleatoriamente una casilla vacía cuando la maquina juega en modo aleatorio.
6. Numpy as np: La libreria "numpy" es una libreria fundamental para la computacion cientifica en Python, proporciona un objeto de matriz multidimensional (ndarray) y funciones para operaciones con estar matrices de manera eficiente, por ejemplo: "np.zero(shape)" devuelve una nueva matriz de forma "shape" y tipo float, llenada con ceros, "np.array(object)" crea una matriz ñampí a partir de una estructura de datos similar (lista, tupla, etc.).
   En el programa se utilizó "numpy" para manejar el estado del tablero de juego con una matriz 2D, facilitando las operaciones de marcado y verificación de casillas.

Manual de usuario

¿Cómo jugar?

El juego consiste en un tic tac toe para poder jugar contra la maquina o contra algún amigo.

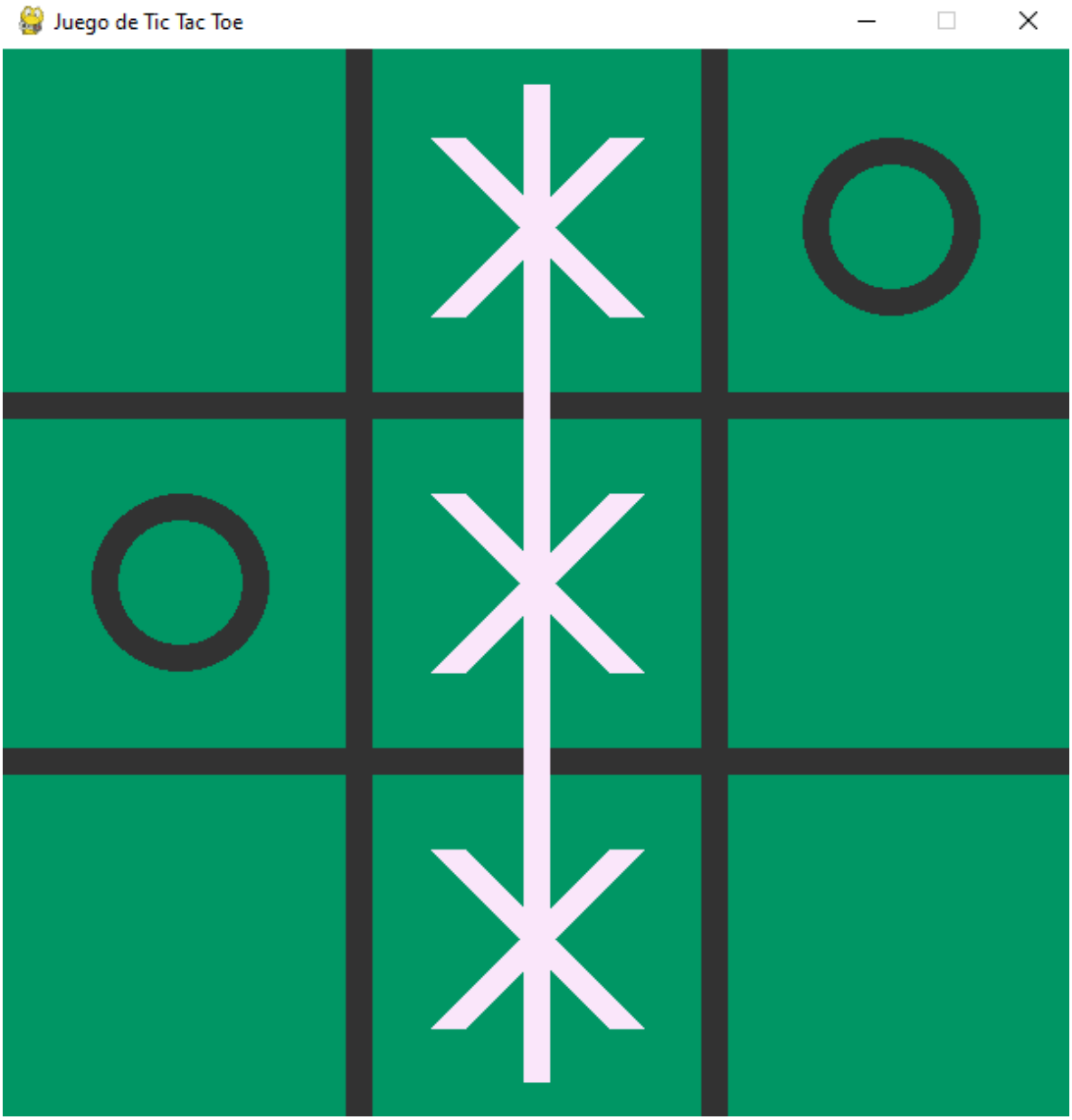
Al ingresar al programa aparece la siguiente ventana



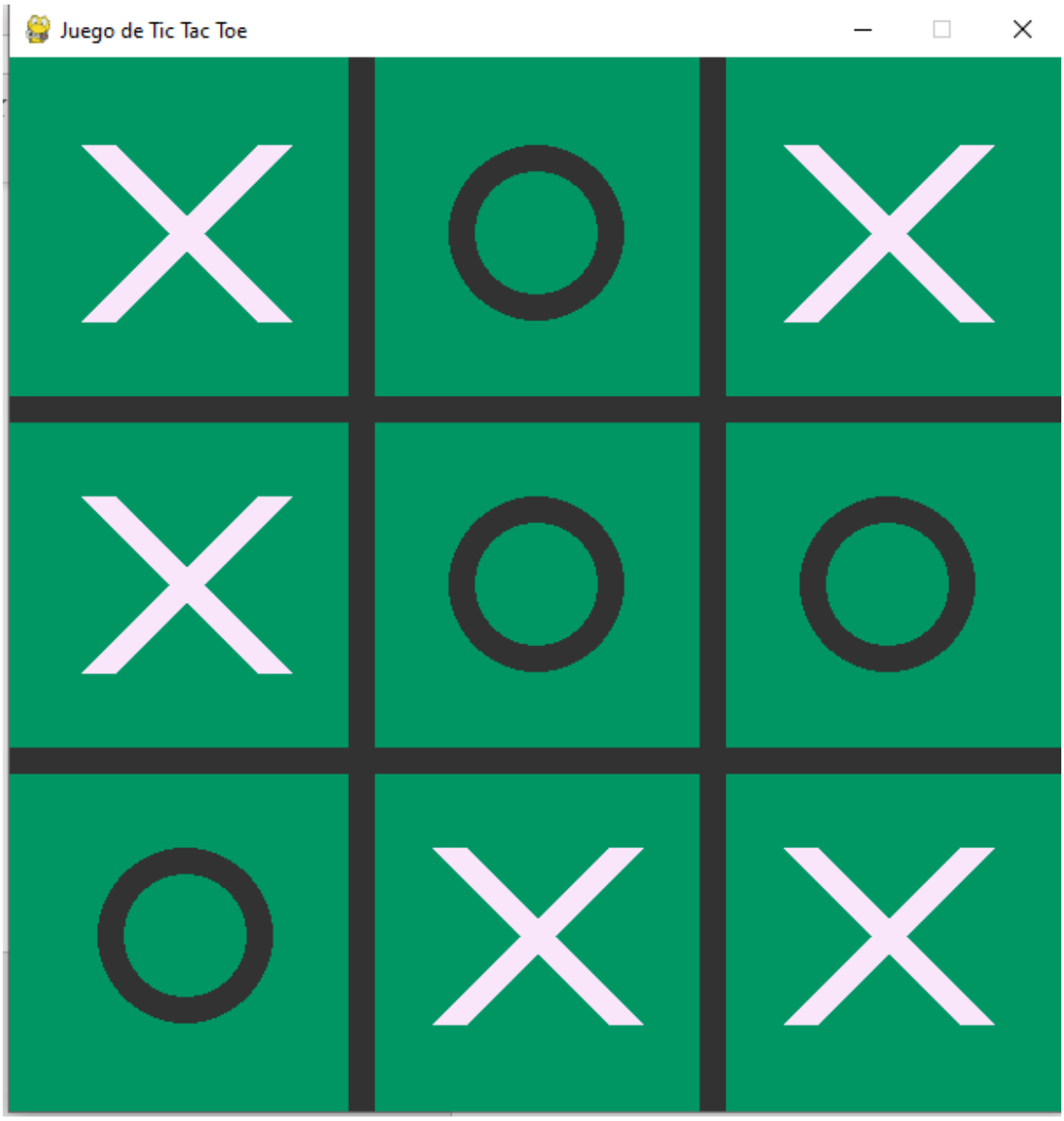
Para jugar contra el nivel básico de la maquina tiene que presionar 0

Al jugar la maquina va colocando círculos en el turno que le corresponde.

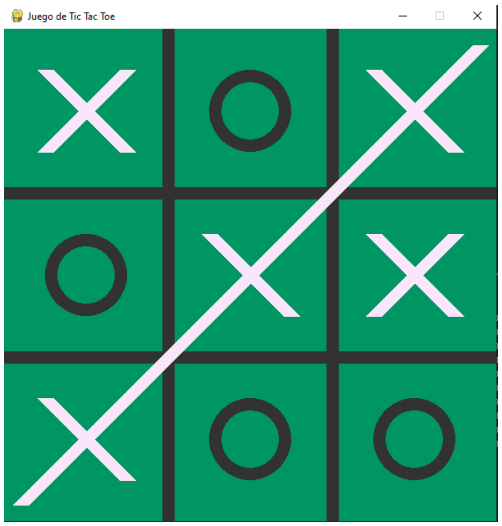
Al ganar se coloca de manera grafica una línea sobre las 3 figuras con las que se ganan de la siguiente manera:

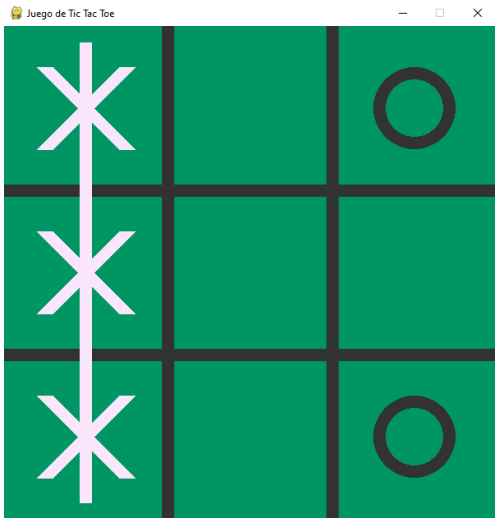


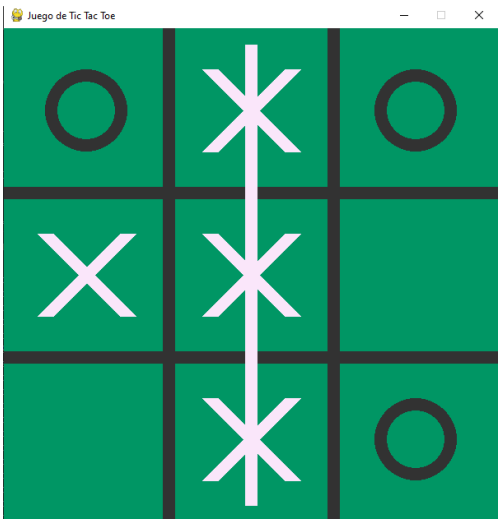
Para jugar de nuevo debe presionar la tecla R y le vuelve a mostrar en limpio el tablero de juego, esto es útil a la hora de empatar como se muestra en pantalla:

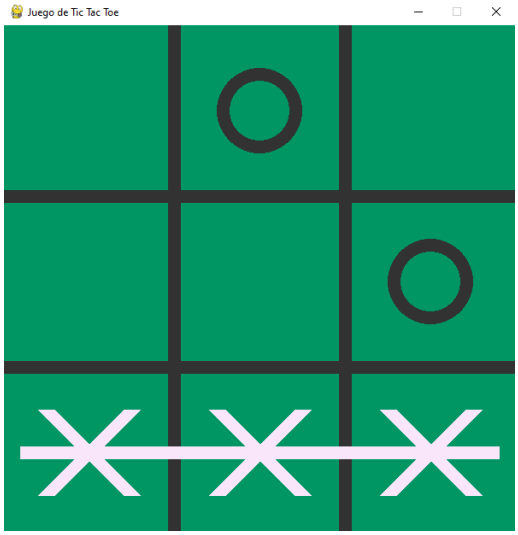


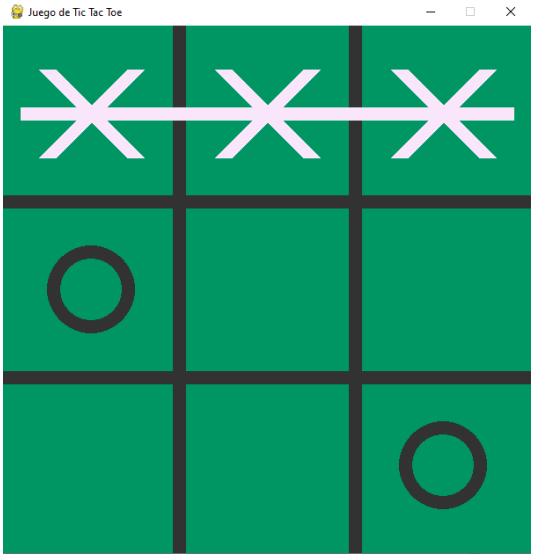
Se puede ganar de muchas maneras como las que se muestran a continuación:

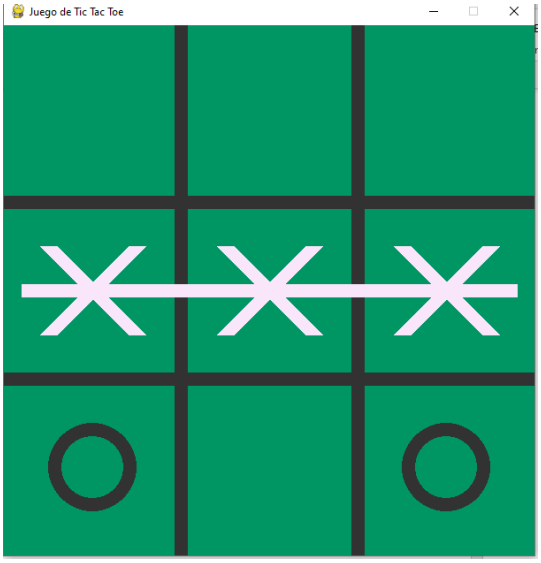


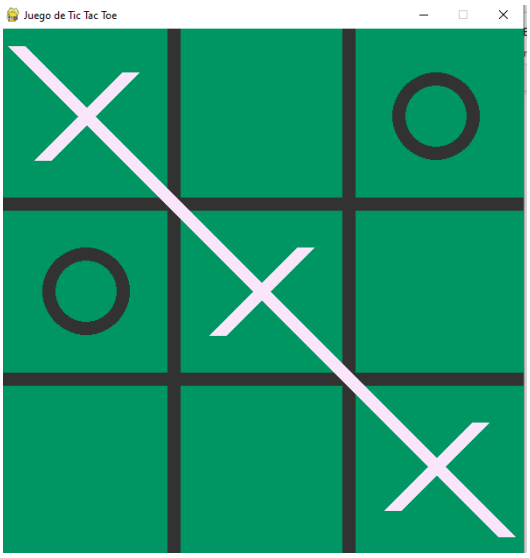


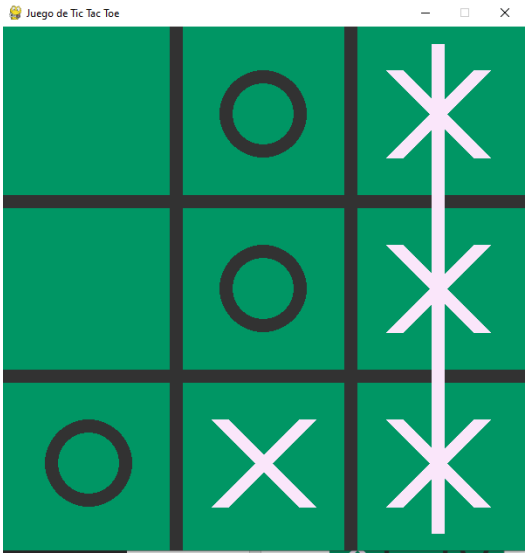












Por último, si desea jugar con un amigo solo presione la tecla G y se deshabilita la maquina y podrá ir presionando cualquier cuadro en la pantalla y se ira variando entre X y O para diferenciar los turnos teniendo en cuenta que siempre inicia X.