

Algoritmos y Estructuras de Datos

Documentación TaCTi

Fecha de entrega: 18/06/2025

Grupo: Bot

Integrantes:

Manzi Peluffo, Camilo Santiago	44.834.897
Pizzi, Matías Nahuel	43.779.349
Módica, Elián Ariel	42.905.229



Organización del repositorio:

En la carpeta tests se realizan las pruebas de todas las funcionalidades que consideren necesario probar “standalone” para luego sumarlo al proyecto.

En la carpeta TaCTi se encuentra la totalidad del proyecto, usando ramas para desarrollo y una rama final de producción. (**main branch**).

Conexión con la API:

Para las peticiones HTTP utilizamos [cURL](#), un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos. Soporta los protocolos FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP, entre otros.

No incluimos la librería dentro de nuestro código, sino que utilizamos la función [popen\(\)](#), que permite ejecutar un comando de shell como si fuera un programa separado.

Para el correcto funcionamiento del programa, debes tener instalado cURL en tu computadora y este debe ser accesible por consola de comandos.

La instalación depende de tu sistema operativo, para Windows se pueden utilizar los gestores de paquetes Winget o Chocolatey (entre otros):

- `winget install curl.curl`
- `choco install curl`

cURL suele estar preinstalado en sistemas basados en Unix, pero también se pueden usar gestores como apt, rpm, yum, etc.

Para verificar que tiene cURL correctamente instalado en su PC, ejecute el siguiente comando por consola:

- `curl --version`

```
$ curl --version
curl 7.84.0 (x86_64-w64-mingw32) libcurl/7.84.0 OpenSSL/1.1.1q (Schannel)
zlib/1.2.12 brotli/1.0.9 zstd/1.5.2 libidn2/2.3.2 libssh2/1.10.0 nghttp2/1
.48.0
Release-Date: 2022-06-27
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ld
aps mqtt pop3 pop3s rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerbero
s Largefile libz MultiSSL NTLM SPNEGO SSL SSPI threadsafe TLS-SRP zstd
```

Para más información, vea la [página oficial](#).

Para convertir los datos en formato legible por la API (JSON), utilizamos la librería [cJSON](#) simplemente agregando los archivos de código y cabecera a nuestro proyecto.

Interfaz Gráfica:

Para la interfaz gráfica se buscó algo con sintaxis simple, fácil de instalar y portable para Windows y Linux.

Raylib es una biblioteca gráfica diseñada para desarrollar juegos 2D en lenguaje C. La elegimos para este proyecto porque permite crear interfaces visuales de forma rápida y clara, sin la complejidad de otras librerías más pesadas como WinAPI o GTK. Además de funcionar con el mismo compilador que utilizamos (Mingw).

Guía para instalación de Raylib:

Descarga desde la página oficial: [Descarga](#)

Para agregar al proyecto en Code::Blocks 20.03 usando Windows:

No es necesario agregar la variable de entorno.

1. Indicar al IDE donde está ubicado el compilador.

Settings → Compiler → Toolchain executables → y colocar la ruta donde está el compilador instalado

C:\raylib\w64devkit\bin

2. Se agrega la ruta al archivo de la librería Raylib (.a) que contiene las funciones ya compiladas.

Project → Build Options → Linker Settings → Link libraries → y colocar la ruta donde esté la librería instalada.

C:\raylib\w64devkit\x86_64-w64-mingw32\lib\libraylib.a

3. Instrucciones al linker para que una el programa con Raylib y las librerías del sistema necesarias para que funcione:

-lraylib: enlaza con Raylib
-lopengl32: usa OpenGL (necesario para gráficos)
-lgdi32: usa GDI (interfaz gráfica de Windows)
-lwinmm: usa WinMM (sonido y multimedia)

Project → Build Options → Linker Settings → Other linker options → y colocar lo siguiente:

-lraylib -lopengl32 -lgdi32 -lwinmm

Frames por segundo

Al ser una interfaz gráfica sin animaciones se puede usar un frame rate relativamente bajo, se probaron distintos valores, pero resulta no funcional con menos de 20 ya que no detecta la presión de los botones.

Organización del juego y turnos:

Ya que no hay un máximo de jugadores por ronda, se debe generar un orden aleatorio de la misma y se debe mostrar como queda el orden de los turnos, entonces, al ingresar los jugadores se guardarán en una lista simple enlazada, se ordenará aleatoriamente y se irá sacando de la lista de a un jugador para encolarlo en la cola destinada para la ronda de turnos. Mediante vayan jugando, se irán desencolando hasta terminar con todas las partidas.

Instrucciones de juego:

Para empezar una partida presione **JUGAR**, aquí podrá escribir el nombre del o los jugadores. Al tipear un nombre si presiona **AGREGAR OTRO JUGADOR** le permitirá seguir agregando más jugadores, o más turnos para el mismo jugador (si se coloca el mismo nombre). Luego de ingresar el/los nombres presione **EMPEZAR** para ir al juego.

Se indicará en pantalla el orden de los turnos. **EMPEZAR** para continuar o salir para agregar más jugadores o volver al menú con **ATRAS**.

Con los turnos asignados, empezarán a jugar uno por uno, indicando previamente si están listos con **COMENZAR** o si quieren abandonar con **RENDIRSE**.

La asignación de símbolos "O" o "X" es aleatoria. Presiona en el tablero en la celda que quieras ocupar. El objetivo del juego es lograr juntar tres simbolos en línea.

Al terminar la partida verás el tablero final y el resultado en la parte superior. Con **SIGUIENTE** se pasa a los siguientes turnos o si es el último jugador, guarda datos de la partida y vuelve al menú.

En el menú está la opcion **RANKING** la cual muestra el ranking online. Si se presiona **RESET** se reinicia.

Mecánica del juego con "IA". El método de los números mágicos.

Cada jugada en el tablero se representa mediante un número mágico que codifica su posición, por ejemplo: 1, 2, 20, 40, 400. El método consiste en recorrer todas las combinaciones posibles de tres jugadas realizadas por un jugador. Para cada trío, se suman los números mágicos y ese resultado se compara contra un conjunto de valores predefinidos que representan las combinaciones ganadoras del juego (las filas, columnas o diagonales completas).

1	2	20	40	400
---	---	----	----	-----

i j k

RESULTADO
23

NO_COMBINA

7	70	111
124	222	421
444	700	

En el gráfico se muestran ejemplos de este proceso. En un caso, las jugadas elegidas son 1, 2 y 20, que suman 23. Este valor no coincide con ninguna de las combinaciones ganadoras, por lo que no hay Ta-Te-Ti. Lo mismo ocurre en otros ejemplos: la combinación de 1, 20 y 40 suma 61 y la de 1, 2 y 400 suma 403, en ambos casos sin formar un Ta-Te-Ti. Finalmente, el ejemplo exitoso es el de 1, 20 y 400, que suman 421. Este valor está en el conjunto de respuestas posibles, indicando que el jugador formó una diagonal y ganó la partida.

1	2	20	40	400
---	---	----	----	-----

i j k

RESULTADO
43

NO_COMBINA

7	70	111
124	222	421
444	700	

1	2	20	40	400
---	---	----	----	-----

i j k

RESULTADO
403

NO_COMBINA

7	70	111
124	222	421
444	700	

Este método permite detectar un Ta-Te-Ti trabajando directamente sobre las jugadas realizadas, sin necesidad de recorrer la matriz del tablero.

1	2	20	40	400
---	---	----	----	-----

i j k

RESULTADO
61

NO_COMBINA

7	70	111
124	222	421
444	700	

Para este trabajo práctico en particular se trató de evitar el uso masivo de IFs para la lógica. El método aplicado sobre listas no es más eficiente para la búsqueda de Ta-Te-Ti que hacerlo directamente en la matriz.

1	2	20	40	400
---	---	----	----	-----

i j k

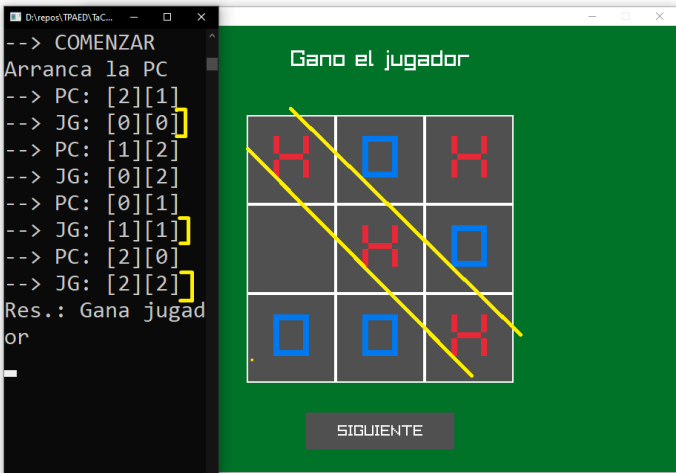
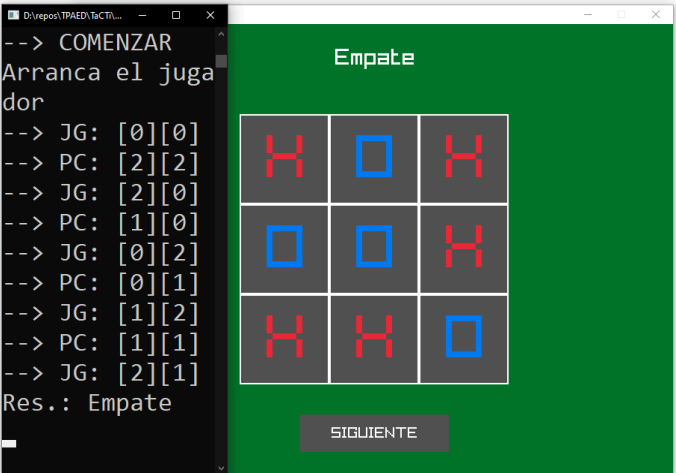
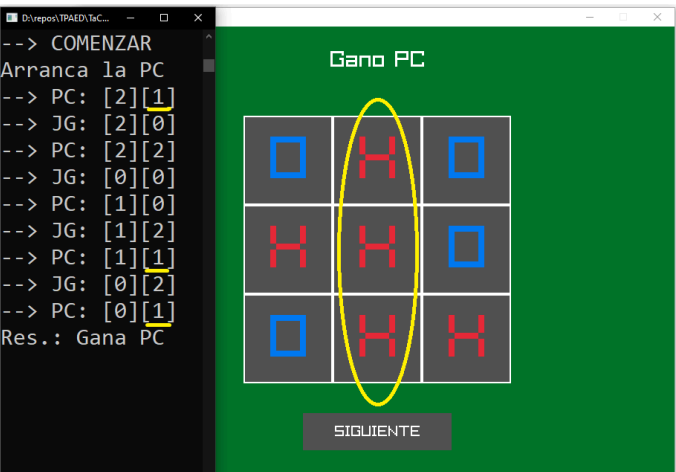
RESULTADO
421

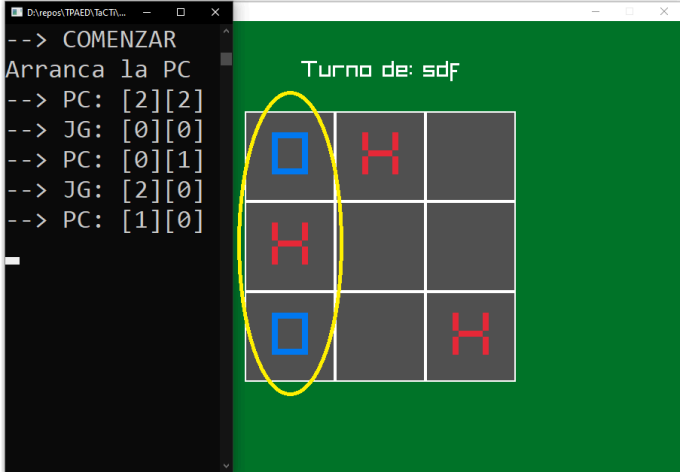
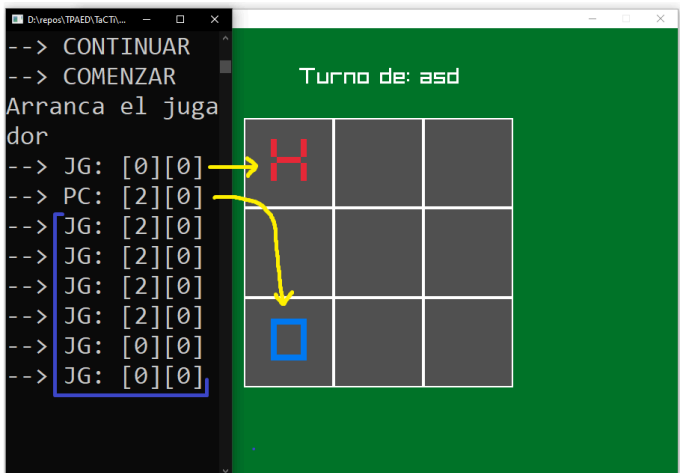
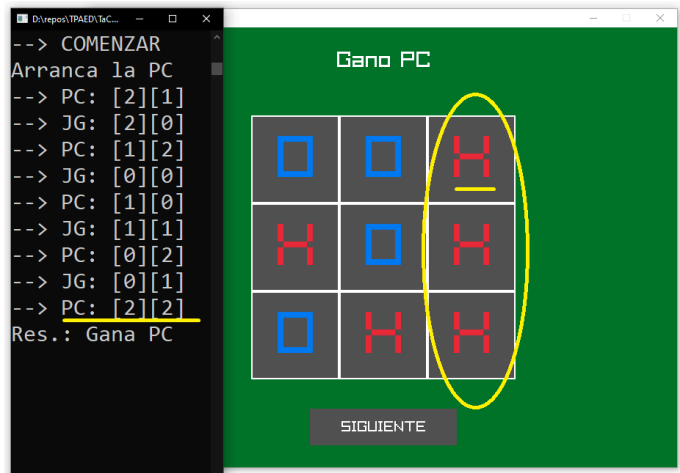
COMBINA

7	70	111
124	222	421
444	700	

Una ventaja que se le encontró a este método es que permite ser ampliado para matrices de NxN.

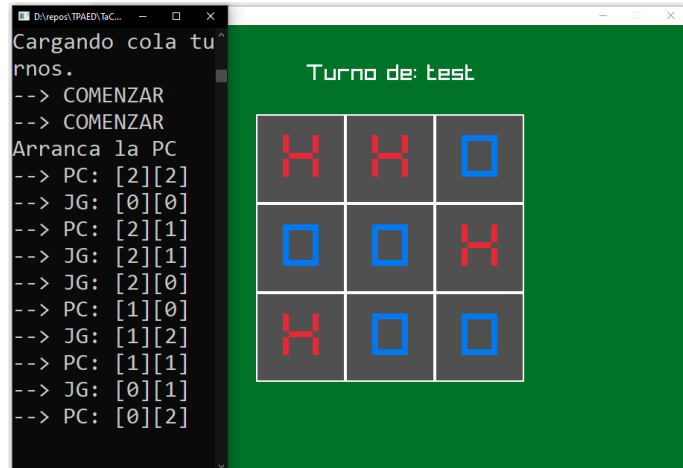
Casos de prueba

Descripcion	Salida esperada	Salida obtenida
El jugador humano gana en diagonal de izquierda a derecha.	Mensaje “Gano el jugador” y siguiente turno o fin.	 <pre> --> COMENZAR Arranca 1a PC --> PC: [2][1] --> JG: [0][0] --> PC: [1][2] --> JG: [0][2] --> PC: [0][1] --> JG: [1][1] --> PC: [2][0] --> JG: [2][2] Res.: Gana jugador </pre>
Se llena el tablero sin que haya un ganador.	Mensaje “Empate” y siguiente turno o fin.	 <pre> --> COMENZAR Arranca el jugador --> JG: [0][0] --> PC: [2][2] --> JG: [2][0] --> PC: [1][0] --> JG: [0][2] --> PC: [0][1] --> JG: [1][2] --> PC: [1][1] --> JG: [2][1] Res.: Empate </pre>
El jugador PC gana con una columna completa (columna 2).	Mensaje “Gano PC” y siguiente turno o fin.	 <pre> --> COMENZAR Arranca 1a PC --> PC: [2][1] --> JG: [2][0] --> PC: [2][2] --> JG: [0][0] --> PC: [1][0] --> JG: [1][2] --> PC: [1][1] --> JG: [0][2] --> PC: [0][1] Res.: Gana PC </pre>

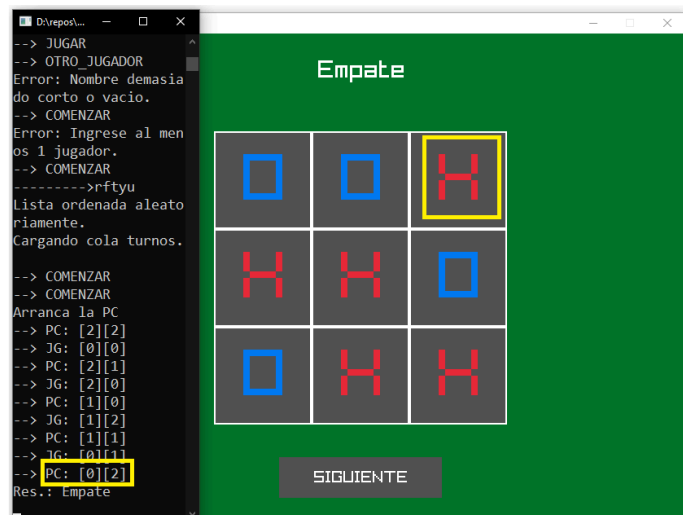
<p>La PC bloquea un posible tatetí del usuario.</p>	<p>Continuar hasta terminar partida.</p>	 <pre> --> COMENZAR Arranca 1a PC --> PC: [2][2] --> JG: [0][0] --> PC: [0][1] --> JG: [2][0] --> PC: [1][0] </pre>
<p>El jugador humano intenta colocar una ficha en una casilla ocupada.</p>	<p>No permitir ingresar en celdas ocupadas. Esperar que ingrese una correcta.</p>	 <pre> --> CONTINUAR --> COMENZAR Arranca el jugador --> JG: [0][0] --> PC: [2][0] --> JG: [2][0] --> JG: [2][0] --> JG: [2][0] --> JG: [2][0] --> JG: [0][0] --> JG: [0][0] </pre>
<p>La PC realiza la última jugada y forma un tatetí en esa jugada.</p>	<p>Colocar simbolo en la posición para ganar Mensaje "Gano PC" y siguiente turno o fin.</p>	 <pre> --> COMENZAR Arranca 1a PC --> PC: [2][1] --> JG: [2][0] --> PC: [1][2] --> JG: [0][0] --> PC: [1][0] --> JG: [1][1] --> PC: [0][2] --> JG: [0][1] --> PC: [2][2] Res.: Gana PC </pre>

La PC realiza la última jugada y completa el tablero sin ganar.

Mensaje “Empate” y siguiente turno o fin.



Algunas veces falló. Se fixeo verificando tablero lleno siempre luego de que juegue la PC.



Luego pruebas varias funcionaron.

El jugador humano gana con una fila completa.

Mensaje “Gano el jugador” y siguiente turno o fin.

