Implementación del Juego 3 en Raya

Adrián Martínez Pérez
Guillem Arnau Vallejos

Facultad de Informática de Barcelona
Grado de Inteligencia Artificial

Índice

1. Introducción

El objetivo de esta práctica es implementar un tablero abstracto para el juego 3 en raya, utilizando el módulo `pygame` de `python3`. Inicialmente, en esta primera versión, hemos desarrollado el juego para ser jugado por dos personas, en lugar de incluir una inteligencia artificial que permita competir contra la máquina, decisión la cual nos ha permitido enfocarnos en el desarrollo de la lógica del juego y en la creación de una buena experiencia de usuario funcional y accesible.

También hemos hecho accesible el código para todo aquel que quiera leerlo, entenderlo o modificarlo, ya que está todo comentado en la medida de lo posible, tanto lo que hacen las funciones, como las diferentes partes de ellas. Además, todo el juego ha sido desarrollado para que pueda ser cambiado el tamaño del teablero y las piedras por jugador, simplemente cambiando el valor de `BSIZ` y `ST_PLAYER` en el archivo `constants.py`, manteniendo una dinámica más entretenida se pretende jugar con un tablero más grande para hacer el juego más largo (y más complicado).

El juego consta de cinco archivos principales que estructuran el código y separan las funcionalidades esenciales:

- `abs_board_h.py`: Funciones y lógica necesarias para gestionar el estado del tablero, verificar condiciones de victoria y manejar los movimientos de los jugadores.
- `constants.py`: Este archivo sirve para almacenar todas las constantes necesarias para el funcionamiento del programa.
- `utils.py`: Implementa las utilidades del juego: tanto las pantallas de carga en el modo texto (`txt`) como en la interfaz gráfica (`gui`), y el menú principal.
- `main_gui.py`: Implementa la interfaz gráfica de usuario. Contiene la configuración inicial, creación de ventanas e integración con `abs_board_h.py`.
- `main_txt.py`: Implementa la versión para consola del juego, con colores y mejoras visuales para un mayor atractivo.

En la carpeta de la práctica se incluye todo lo necesario para la ejecución y disfrute del juego. Solo es necesario ejecutar dos archivos para jugar al juego:

- Para jugar en modo texto, ejecutar en la consola el siguiente comando:

```
$ python3 main_txt.py
```

- Para jugar en modo gráfico, ejecutar en la consola el siguiente comando:

```
$ python3 main_gui.py
```

2. Metodología de la Implementación

El desarrollo de este proyecto ha sido un trabajo colaborativo entre Adrián Martínez Pérez y Guillem Arnau Vallejos, mediante un repositorio compartido en la plataforma `GitHub`. Esta plataforma nos permitió realizar un seguimiento de los cambios, revertir errores y garantizar la estabilidad del código en cada etapa del desarrollo. GitHub ha sido una excelente plataforma para la colaboración en este tipo de proyectos, y hemos disfrutado mucho aprendiendo a utilizarla.

En cuanto a la distribución de tareas, adoptamos un enfoque flexible. No dividimos las tareas de manera estricta, sino que trabajamos de forma iterativa comunicándonos constantemente sobre los avances, problemas encontrados, mejoras implementadas y las ideas para futuras versiones. Esta dinámica fomentó un ambiente de aprendizaje mutuo y creatividad.

Para la implementación de este código, hemos añadido un archivo que no estaba originalmente: `utils.py`, y hemos modificado todos los archivos con el fin de lograr una buena implementación, teniendo creatividad además de añadir nuevas funciones y utilidades, como una animación de carga para el modo de juego txt.

3. Modos de juego

4. Explicación del Código

Para poder explicar todo el código sobre como funciona nuestra implementación del juego del tres en raya, debemos de explicar todos los archivos que hemos modificado `abs_board_h.py, constants.py, main_gui.py y main_txt.py`, además del archivo de nueva creación que hemos incluido nosotros: `utils.py`.

4.1. abs_board_h.py

Inicialmente, en este archivo solo se nos dierno nombradas las funciones que debíamos de utilizar, dejándonos a nosotros una libre implementación del código. La función principal es set_board_up, la cual inicializa el tablero, y dentro de esta función tenemos la lógica de todo el tablero. Modificamos todas y cada una de las funciones para implementar la lógica del juego, y hacer que fuera sencillo de entender.

En este archivo se inicializan todas las variables necesarias, como el tablero (con medida *BSIZ*, definida en constants.py, y que permite que se pueda cambiar el tamaño del tablero), las piedras jugadas y las no jugadas en forma de una lista, el jugador actual al que le tocaría mover piedra, y la piedra seleccionada en el turno actual. A continuación, se explican cada una de ellas:

- `stones`: Retorna un iterador sobre las piedras que ya se han colocado en el tablero.
- `select_st`: Selecciona una piedra del tablero para moverla, y toma como parámetros i y j, las cuales son las coordenadas de la piedra a seleccionar, y retorna True si la selección es exitosa, o False si la selección no es válida.
- `end`: Comprueba las condiciones de victoria y muestra el ganador. Además, esta función es clave para el funcionamiento del modo de juego *misery*, ya que incluye una comprobación para el modo de juego, el cual invierte al ganador.
- `move_st`: Coloca una nueva piedra en el tablero, hasta que se acaban, y entonces cambia su comportamiento para mover la piedra seleccionada. Al igual que la anterior, recibe las coordenadas de la piedra. Esta función retorna una tupla que indica si el movimiento fue válido, el jugador actual y si el juego ha terminado.
- `draw_txt`: Dibuja el estado actual del tablero cuando está seleccionado el modo txt, y hemos realizado mejoras visuales para que el tablero se vea atractivo con el uso de símbolos ASCII.

4.2. constants.py

Dentro de este archivo se definen todas las constantes del juego, haciéndolo así más modificable. Incluye el tamaño del tablero, las piedras por jugador, los colores del juego para el modo gui, además del tamaño de la ventana, celdas vacías, tamaño de las piedras... Además, nos sentimos libres de añadir un nuevo diccionario llamado `COLORES`, dentro del cual se definen nuevos colores como el amarillo, violeta, magenta o cyan. Todo esto con el fin de modificar los colores en el modo txt, que se impriman las piedras con diferentes colores, o incluso la animación de carga. Gracias a este diccionario, se pueden modificar los colores fácilmente, y son accesibles para todos los archivos python, e incluso podríamos añadir colores a los logs de cada archivo para detectar posibles fallos más fácilmente.

Pensamos en utilizar en este código el formato de secuencia de escape ANSI, ya que con esta codificación es más fácil modificar el texto, como podría ser ponerlo en negrita o cursiva, en lugar del formato tradicional RGB.

4.3. utils.py

Este archivo no estaba inicialmente en el código proporcionado, pero lo hemos considerado muy útil para implementar diversas funcionalidades, como el menu de inicio o las pantallas de los ganadores.

Primeramente, la primera función en este archivo es `draw_winner_board`, la cual implementa la pantalla del ganador del juego. Esta función acepta como parámetro a *winner*, el cual es el ganador del juego. Primeramente incluye una comprobación para verificar el archivo que ha sido ejecutado (main_gui.py o main_txt.py), con el objetivo de mostrar una interfaz u otra. En este caso, si el modo de juego seleccionado es txt, se llama a la

función `draw_winner_txt` la cual es la encargada de dibujar el ganador en el modo txt, y se le pasa el parámetro winner.

Esta función renderiza una nueva ventana utilizando el módulo `pygame`, y se encarga de renderizar un texto u otro en base al ganador (pasado como winner a la función), y dependiendo del modo de juego.

Posteriormente, hemos implementado la función `draw_winner_txt`, la cual se encarga de dibujar la pantalla del ganador en el modo txt. Esta función recive como parámetro el ganador del juego, y se imprime el texto en consola en diferentes colores utilizando el diccionario anteriormente mencionado para un mayor atractivo visual.

La última función en este archivo es `show_menu`, la cual fue todo un reto de programación. Esta función implementa el menú principal del juego, y por ello es una de las más importantes.

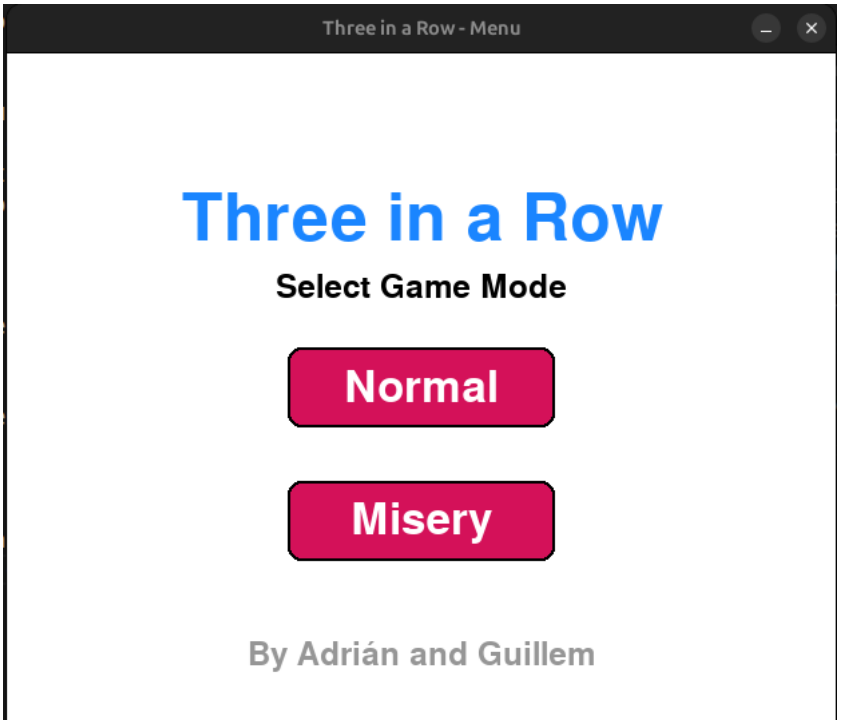

Figura 1: Menú principal del juego en el modo gui (Graphical User Interface).

Dibuja una ventana con el título del juego, y con la posibilidad de seleccionar el modo de juego 'Normal' o 'Misery', con dos atractivos botones rojos y con un hover azul, el cual se muestra al pasar el ratón por encima, y ayuda a diferenciar el modo de juego que se va a seleccionar, además de aportar un toque dinámico al menú. Debajo se muestran los autores del juego.

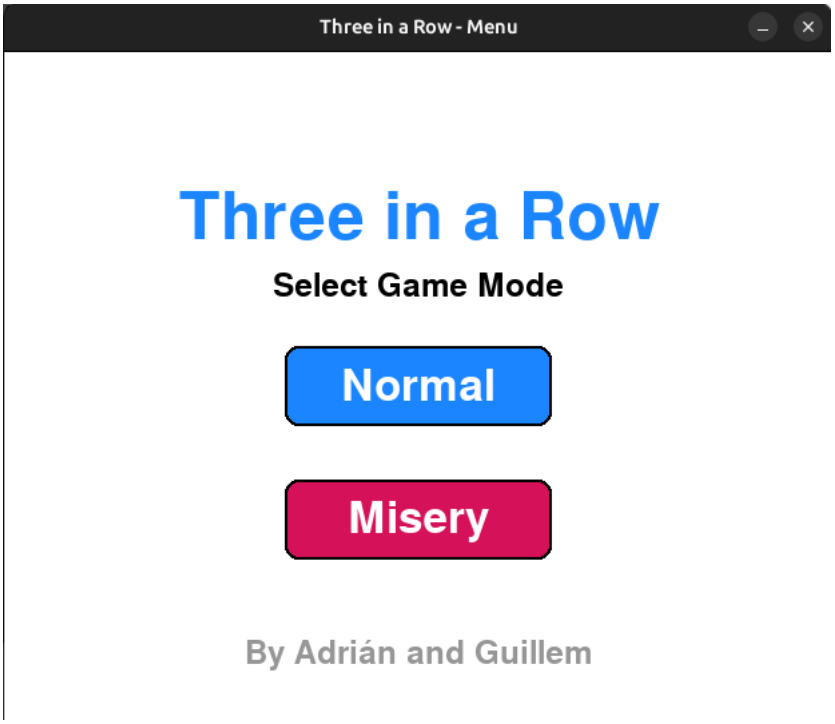

Figura 2: Hover azul en el menú principal del juego.

4.4. main_gui.py

Este archivo de Python es el más importante para implementar el menú del juego en modo GUI (Graphical User Interface), y el cual ofrece una interfaz de juego más amigable que la tradicional consola. Es sencilla de utilizar, se puede jugar con simples clicks y se visualizan correctamente todas las piedras, aunque la funcionalidad sigue siendo la misma que en consola. Está implementado de forma que se muestra una ventana de tamaño variable en función del tamaño del tablero, y en su fase inicial, muestra casillas en blanco indicando donde puede ser una piedra colocada, y un rectángulo el cual indica el color del jugador al que le toca colocar una piedra.

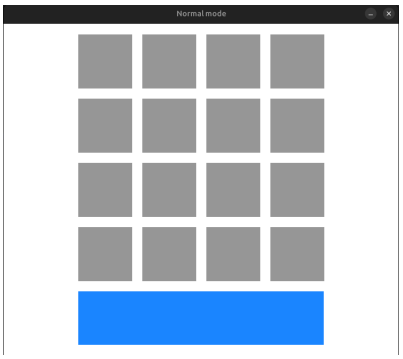

Figura 3: Ejemplo de tablero 4×4.

Como podemos observar en la figura 3, podemos modificar el tablero a nuestro gusto, así como las piedras de las que dispone cada jugador (para un tablero $N \times N$, cada jugador tendrá $\frac{N^2-1}{2}$ piedras (siempre que $N^2 - 1$ sea par)), y hacer el juego aún más interesante.

Dentro de este archivo hemos definido varias funciones vitales para su correcto funcionamiento:

- `trans_cord`: Traduce las coordenadas de los píxeles de la ventana pygame en coordenadas del tablero.
- `draw_square`: Dibuja un cuadrado en la pantalla en las coordenadas específicas del tablero, sirve para mostrar los cuadrados (tanto vacíos como con piedra). Se le introduce como valor el tamaño de la pantalla, y las coordenadas, y utiliza la función `pygame.draw.polygon` para conseguir esto.
- `draw_stone`: Sirve para dibujar la piedra en la pantalla en las coordenadas especificadas que se le introducen a la función, así como el color en función del jugador al que le toque el turno. Utiliza la misma función antes mencionada pero con un círculo (`pygame.draw.circle`).
- `draw_board`: Es la función principal, y se encarg de dibujar el tablero completo, utilizando las funciones anteriormente mencionadas. Dibuja las casillas en horizontal y vertical, y todas las piedras que sean necesarias. Acepta el parámetro `curr_player`, para saber cual es el jugador actual y dibujar un rectángulo del color de dicho jugador, y end, para saber si debe de seguir dibujando tablero o el juego ya ha acabado.

Para seleccionar el modo de juego, dentro de este mismo archivo, y antes de ejecutar las funciones que se encargan de ejecutar el juego propiamente dicho en el modo seleccionado, hemos agregado una variable llamada `selected_mode`, la cual llama a la función `show_menu`, que invoca el menú principal del juego, y mediante los botones que se han mostrado anteriormente en el menú principal retorna `'normal'` si se ha seleccionado el modo normal o `'misery'`, si se ha seleccionado el modo misery.

Posteriormente se han definido las funciones `normal_mode` y `misery_mode`. Hemos optado por definir dos funciones diferentes para cada modo de juego para poder modificar el comportamiento de cada modo de juego si fuese necesario sin que este afecte al otro, además facilitar la legibilidad del código.

Si se ha seleccionado el modo normal, se ejecuta el juego en modo normal, y en caso contrario se ejecuta el juego en modo misery.