**Instituto Tecnológico de Costa Rica**

**ITCR**

**Escuela de Ingeniería en Computación**

**Profesor:**

**Diego Mora**

**Tarea Programada II:**

**Conecta Cuatro con Tablero Infinito**

**Semestre I**

**2019**

**Curso:**

**Taller de programación**

**Grupo:**

**5**

**Estudiante:**

**Ignacio Álvarez Barrantes**

**2019039643**

**Tabla de Contenidos**

1. **Introducción 3**
2. **Descripción del problema 4 - 8**
3. **Análisis de resultados 9 - 10**
4. **Conclusiones 18**
5. **Estadísticas de tiempos 19**

Introduccion:

Para la segunda tarea programada se requiere la elaboración de un cuatro en linea. A continuación se da una breve explicación de en qué consiste dicho juego:

***“Cuatro en línea***

*Cuatro en línea es un juego con el objetivo de conseguir colocar cuatro fichas de un modo adyacente tanto vertical, horizontal o diagonalmente, evitando en todo momento que el rival sea capaz de recrear anticipadamente tu objetivo.”*

Dicho juego consta de una matriz de 6 filas y 7 columnas, un total de 42 posibles posiciones para jugar. Además de 2 fichas de colores diferentes que representan a los jugadores. Cada jugador coloca una ficha a la vez, funcionando como turnos. La idea es emular este juego de mesa en Python mediante el uso de la librería Pygame para así poder representar de forma gráfica el juego, con el fin de que este sea más amigable hacia el usuario. En cuanto al tablero, las reglas difieren un poco ya que los jugadores serán capaces de extender el tablero de forma indefinida (a la derecha, izquierda y hacia arriba). El tablero deberá contar con indices para así informar al jugador en todo momento en cúal sector del tablero se encuantra ubicado.

Descripcion del problema:

**Cuatro en línea**

Cuatro en línea es un juego con el objetivo de conseguir colocar cuatro fichas de un modo adyacente tanto vertical, horizontal o diagonalmente, evitando en todo momento que el rival sea capaz de recrear anticipadamente tu objetivo.



El juego original consta de una matriz de 6 filas y 7 columnas y 42 fichas de dos colores distintos, las cuales se reparte en igual cantidad a cada uno de los dos jugadores posibles.

Cada jugador coloca una ficha a la vez, en turnos intercalados. Es posible que el juego quede empatado. *Note que esta posibilidad solo aplica para el juego de mesa original, al poseer un tablero infinito el juego no puede quedar en empate.*

**DESCRIPCIÓN**

Para el segundo proyecto programado debe implemetar en Python un juego de 4 en línea, sin embargo, difiere en algunas características del juego original. A continuación se explican las características del proyecto:

El objetivo del juego, la cantidad de jugadores (2), la forma de los turnos entre jugadores, se mantienen.

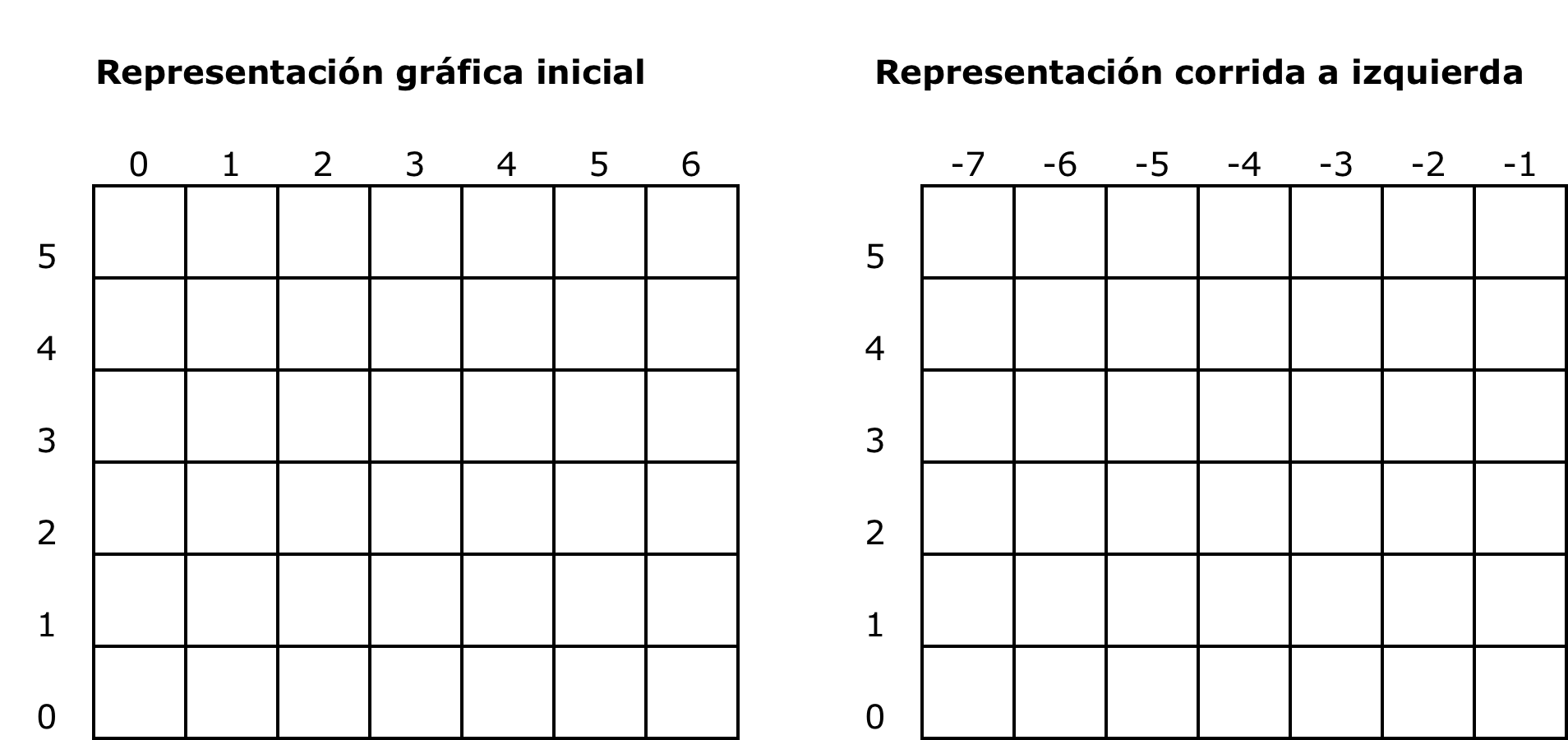
La matriz no será de 6X7 únicamente, sino que es infinita. La matriz inicia con 6x7, pero los jugadores podran colocar la ficha entre las 7 columnas o bien, a la derecha o izquierda 7 posiciones de cualquier ficha y la matriz lo permitirá, pues es infinita hacia los lados, pero con esa restricción, solo se puede colocar fichas en columnas que estén a una distancia máxima de 7 columnas de una ficha.

Hacia arriba la matriz también es infinita.

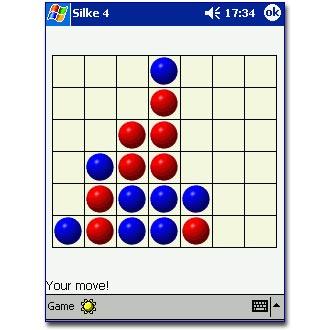
1. **Representación gráfica:**

En pantalla siempre debe verse 6X7 fichas y tendrá que implementar un “scroll“ para ver la matriz si ha crecido, sin embargo, para más facilidad, solo se verán 6x7 en todo momento.

Importante es que las columnas y filas deben estar siempre numeradas, para saber en todo momento en cuál posición de la matriz se está.



El programa debe ser gráfico, mostrar colores de fichas según el estado de la matriz.



También debe mostrar cuál es el jugador que le corresponde el turno.

1. **Jugadores:**

Solamente pueden participar dos jugadores, donde cada uno tendrá un color de ficha. Los turnos son intercalados. En la pantalla debe mostrarse turno.

Antes de iniciar el juego, debe solicitarse el nombre de ambos jugadores.

1. **Juego contra la computadora:**

Puede jugar un solo jugador contra la computadora. Este modo de juego tendrá que implementar un algoritmo que haga el segundo turno. Debe ser un algoritmo que se base en colocar la ficha donde no se pierda, es decir, donde el otro jugador no pueda formar cuatro en fila. Cuidado porque habrá ocasiones en que no hay alternativa y se le podría “enciclar“ el algoritmo.

1. **Puntajes:**

Debe tener un archivo donde guarde los puntajes (cantidad de juegos ganados) de todos los jugadores del juego. Esto adicional a una pantalla donde se puedan consultar los puntajes, ordenados del mayor a menor.

1. **Guardar partida:**

Debe permitir guardar una partida no finalizada y luego poder jugarla nuevamente en el estado en el cual quedó. Para esto utilice archivos para almacenar en disco las partidas.

1. **Pygame:**

Se importará una librería para la implementación de la interfaz gráfica, esta es de elección del estudiante. En este caso se utilizará la librería Pygame. La cual se puede obtener por medio de la página :

[*https://www.pygame.org*](https://www.pygame.org/)

Dicha librería es de uso gratutito y de bajo nivel, reuqiere un poco de lectura para poder comprender sus componentes y funciones incluidas. Dentro de la documentación interna se incluyen comentarios que indican la función de ciertas funciones específicas de la librería, las cuales se pueden encontrar en el siguiente link:

[*https://www.pygame.org/docs/index.html*](https://www.pygame.org/docs/index.html)

**Análisis de resultados**

Esta fue la primera experiencia en cuanto a la elaboración de un videojuego, por lo cual no se tenía suficiente experiencia con respecto a diversos aspectos que conllevan la elaboración de un programa con dicho objetivo. Todos estos pequeños problemas se fueron haciendo notorios conforme se iba avanzando en la elaboración del código.

Empezando con el problema primordial. ¿Cúal herramienta usar para la elaboración de la interfaz gráfica del juego? Ya que la única experiencia que se poseía previo a la elaboración de la tarea programada era Tkinter. Aunque está fuese una librería fácil de usar para la elaboración de interfaces gráficas, esta además tenía ciertas limitaciones que forzaban el programa a verse robusto y no muy atractivo al usuario. Por lo que se decidió investigar nuevas librerías para remplazar a Tkinter. Aquí fue donde apareció Pygame. Una librería hecha con el fin de elaborar juegos bajo un sistema de construción de lenguaje Python. En un principio la librería fue un poco difícil de utilizar ya que funcionaba por medio de eventos (como movimiento del mouse, presionar el mouse o el teclado). Enteder el funcionamiento base de la librería resultó ser de las partes más complicadas y extensas del trabajo. Ya que se debía empezar desde 0, además de que dicha librería es considerada de alto nivel. Una vez comprendido la mayoría de las funciones básicas ,la elaboración del juego fue relativamente sencilla.

A su vez, se presentó otra incógnita, que aunque no fuera relacionada a la estructura lógica del juego, era de vital importancia para el mismo. ¿Cúal temática iba a tener el juego?. Aunque fuese un conecta 4 normal, debía tener una interfaz llamativa para el usuario y de fácil comprensión. Debido a esto fue que se optó por una tématica “Retro Neon”, algo similar al estilo de las películas “Tron”, así fue como se eligieron los diseños como los fondos de las pantallas, la fuente de la letra y las canciones que suenan de fondo.

Una vez iniciada la construcción del código y de la interfaz se presentaron nuevos problemas, como la integración de una algoritmo que permitiera jugar solitario, es decir implementar un algoritmo que fuese capaz de realizar el segundo turno. Para esto se utilizó el algoritmo usado para detectar el gane y así crear uno que detectara un posible gane y en base a esto retornara la columna faltante para que así la computadora pudiese bloquear una posible jugada ganadora.

Luego la invención de una forma para poder guardar una partida y los jugadores del Ranking, aunque se tenía una idea vaga de como guardarlos, gracias a las funciones dadas por el profesor para la tarea programada anterior. Se contaba con una nueva dificultada y era idear la mejor manera para almacenar la información requerida. Al final se optó por un sistema similar al usado en programas anteriores, mediante el uso de listas.

Y por ultimó durante la elabortación del menú gráfico, se evidenció una gran desventaja del uso de Pygame. Aunque pygame es una librería para montar juegos de forma gráfica no es muy eficiente con respecto a la elaboración de menús. Ya que no presnetaba formas sencillas de implementar botones e inputs para así hacer la navegación dentro del juego mucho más sencilla. A diferencia de Tkinter que con una simple función podía realizar ambas cosas. Por un momento se consideró la implentación de ambas librerías para asi facilitar la creación de este tipo de elementos. Pero esto demostró no ser de gran ventaja ya que generaba confusión en el código y du edtructura. Por lo que se recurrió de nuevo a la busqueda de información para encontrar una forma “sencilla ”, para implementar algoritmos similares en pygame.

La página [*https://stackoverflow.com*](https://stackoverflow.com/)fue de gran utilidad para la comprensión de la librería de donde se obtuvo los códigos para el input:

[*https://stackoverflow.com/questions/70797/user-input-and-command-line-arguments*](https://stackoverflow.com/questions/70797/user-input-and-command-line-arguments)

Y el código para la elaboración de botones en la interfaz:

[*https://stackoverflow.com/questions/40336960/creating-rect-buttons-with-pygame*](https://stackoverflow.com/questions/40336960/creating-rect-buttons-with-pygame)

Aunque en la página se econtraba el código no fue simple copiar y pegar ya que el código debía ser adaptado al resto del código. Lo cual requirió aún más búsqueda para así comprender el funcionamiento de las funciones usadas y como implementarlas al programa.

En conclusión fue una proyecto que requirió de tiempo y mucha búsqueda, mas una vez concluido genera una gran satisfacción.

**Estadística de tiempos**

|  |  |
| --- | --- |
| **Análisis de requerimientos** | **3h** |
| **Diseño de la aplicación** | **66h** |
| **Investigación de funciones** | **22h** |
| **Programación** | **15 h** |
| **Documentación interna** | **4h** |
| **Pruebas** | **10h** |
| **Elaboración de documento** | **1.5h** |
| **TOTAL** | **121.5h** |