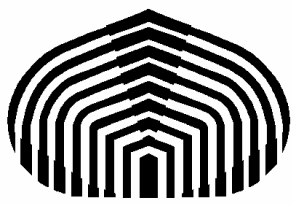
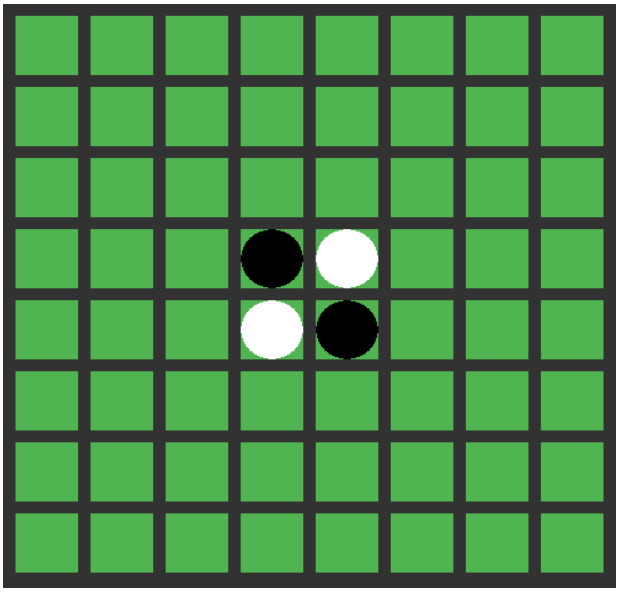


UNIVERSIDAD SIMÓN BOLÍVAR
COORDINACIÓN DE INGENIERIA EN COMPUTACION
CI2691: LABORATORIO DE ALGORITMOS Y ESTRUCTURAS I

OTHELLO



Profesor: Carlos Perez

Kevin Briceño 15-11661
Ángel Rodríguez 15-11669

INTRODUCCIÓN

El juego de Reversi, o también conocido como Othello por su publicación por Mattel, es un juego de mesa de estrategia que se realiza en un tablero 2D de dimensiones 8x8; con 64 fichas de 2 caras y colores distintos para cada cara, generalmente se emplean negro y blanco. En cada una de las casillas se colocará una pieza ya sea en su lado negro o blanco, la cual determinará cuál de los jugadores posee esa casilla. El objetivo del juego es terminar con más casillas de tu color. Los jugadores toman turnos colocando fichas en el tablero, las cuales, cuando estas son jugadas, todas las fichas del otro color que se encuentren entre la ficha jugada y una ficha del mismo color, se convertirán en el color de la jugada realizada.

Reversi está implementado con 2 componentes básicas; una de cálculos que contiene la parte lógica del juego y una de interfaz.

El juego está realizado en lenguaje de programación Python. Con la ayuda de análisis descendente, el programa principal, logra tener una lectura de código ordena y menos extensa, lo que permite la identificación detallada y rápida de un fragmento del código total y una interpretación más rápida para quienes deseen visualizar el código.

Pygame es una librería para Python que implementa un motor gráfico para la creación de un programa interactivo; con ella se realizar toda la interacción usuario-maquina necesaria para la ejecución del archivo. Hace más amigable la ejecución y lo impreso por la pantalla para los programadores y, aún más, para los no programadores.

DISEÑO

Los siguientes archivos contienen funciones resultado del análisis descendente realizado para dividir problemas más grandes en subproblemas más pequeños; también conocido como "*divide and conquer in combine*".

- **Funcion0_Nombre**: con este archivo, le solicitamos a los usuarios que ingresen sus nombre o alias de juego; además elige, aleatoriamente, quien será el jugador en iniciar la partida, es decir, quien será el jugador de fichas negras. Note que los datos ingresados deben poseer al menos un (1) carácter. La función contenida en este archivo, tiene 2 parámetros de entrada: player1 y player2 respectivamente.
- **Funcion1_SePuedeJugar**: mientras el número de fichas en el tablero sea mayor que cero y esta función dentro del archivo retorne verdadero, se juega un turno más. En particular, la función determina si alguno de los dos (2) jugadores queda sin fichas de su color en el tablero; en caso de que eso ocurra, retornará "False". Al igual que la anterior, posee dos parámetros de entrada: *player1 y player2*; respectivamente.
- **Funcion2_Turno**: determina el jugador en turno del momento. Sus parámetros de entrada son tres (3): *esTurno, player1 y player2*; respectivamente. El parámetro "*esTurno*" es un número entero que, mientras sea impar, arrojara el turno para el player1 y en caso contrario, arroja el turno a player2.
- **Funcion3_EsValida**: contiene simultáneamente dos (2) funciones principales y ocho (8) secundarias. La primera con el nombre *"ListaJugadasValidas*" hace un recorrido por todo el tablero para determinar si el jugador en turno tiene alguna posibilidad de jugada, en caso contrario arrojará un *"False"* cambiando el turno y el jugador, sus parámetros de entrada son: *array y player*; respectivamente; la segunda función denominada "*JugadaValida*", verifica que la coordenada dada por el jugador en turno sea correcta para poder realizar una jugada en caso contrario, imprime un error o pasa, los parámetros de entrada para esta función son: *tablero, cord1, cord2, y player*; respectivamente. Las otras funciones verificaran de forma vertical, horizontal y/o diagonal, las condiciones para que una jugada sea válida.

- **Funcion4_juega**: tiene una (1) función principal y ocho (8) secundarias. La primera es la encargada de realizar la jugada en la coordenada dada por el jugador en turno. Similarmente a la función anterior, las secundarias buscan en 8 direcciones del tablero y, además, cambia las líneas correspondientes a una jugada correcta. La principal recibe cuatro (4) parámetros de entrada: *tablero, cord1, cord2, y player*; respectivamente. Y las secundarias también reciben la misma cantidad de argumentos: *board, f, c y jug;* respectivamente. Básicamente son los mismos parámetros de entrada, pero se nombran distintos para evitar algún problema con las variables globales.
- **Funcion5_Casillas**: la función contenida en este archivo realiza un recorrido por todo el tablero y va sumando un contador si encuentra una casilla perteneciente al jugador designado. Los parámetros de entrada son: *tablero y player*; respectivamente.
- **Funcion6_resultado**: imprime en pantallas cuantas casillas posee cada jugador de su color en ese turno. Sus parámetros de entrada son: *player1, y player2*; respectivamente.
- **Funcion7_Total**: compara los resultados de las casillas y determina quién gana la partida. Sus parámetros de entrada son: *player1 y player2*; respectivamente.

IMPLEMENTACIÓN

El programa consta de una serie de variables inicializadas, algunas para la parte lógica y otras para la interfaz. Las variables son:

- "jugador1" y "jugador2": están relacionada con una clase, que posteriormente se va a definir, para almacenar los nombres de los jugadores, su identidad en el tablero, y la cantidad de casillas que poseen cada uno
- "X" y "Y": representan las coordenadas en el tablero introducidas por los jugadores, para filas y columnas respectivamente.
- "tablero": es un arreglo bidimensional de 8x8, en él se aplicarán las jugadas dadas por los usuarios.
- Se determinó el número "0" para indicar que las casillas están vacías, el "1" para indica que la casilla pertenece al jugador1 y el "2" representado para jugador2. Además, "1" y "2" están considerados como fichas negras y fichas blancas respectivamente.
- La variable "ficha" lleva en conteo regresivo de cuantas fichas se han jugado y cuantas quedan por jugar.
- "jugador" es una variable que depende de "jugador1" y "jugador2" para indicar de quien es el turno en ese momento, si puede jugar y en dónde puede hacer.
- "turno" es un contador ascendente inicializado en 1 que mediante la paridad determina que jugador realiza la acción.
- "otra" es un string, una vaiable de entrada que solo recibe SI o NO para determinar los condicionales a una nueva partida
- "BLACO, VERDE, GRIS, FONDO, NEGRO" son las inicializaciones de las variables de los colores en rgb.
- "LARGO, ALTO, MARGE, DIMENSION" variables para las dimensiones de ventana de la interfaz.
- ventana: crea la ventana principal
- clock: crea un reloj para refrescar la ventana
- fuente: para visualización de las letras
- color y colorf: determinan el color de las ventanas y las fichas de cada jugador

Se genera una clase de nombre "jugador" en la cual se va a almacenar el nombre o alias de los jugadores, su identidad en el tablero y la cantidad de casillas que han convertido cada uno de su color.

La función llamada partida es la principal, permite llamar al resto de las funciones definidas y hacer la ejecución del juego, tanto lógico como gráfico. Luego de la llamada a la función principal, inicia el "pygame.init()" que permitirá abrir la ventana de interfaz, cuyas dimensiones y colores vienen asignadas posteriormente; así mismo, se inicializa el tablero, las primeras posiciones de la fichas de este, la cantidad de casillas que posee cada jugador al comienzo(2 para cada uno) y la cantidad de fichas disponibles al comienzo (total 60 fichas). Seguidamente, se abre el bucle "while" y verifica los eventos de Pygame con un ciclo "for". Mientras el usuario no cliquee la opción de cerrar en la ventana, se evaluarán las siguientes condicionales:

1. si la cantidad de fichas totales es mayor que cero y la cantidad de fichas del color de algunos de los jugadores es distinto de cero, entonces se puede realizar una jugada. En caso contrario, sale de la partida dando los resultados del ganador y, luego, preguntara a los jugadores si desean jugar otra ronda.
2. Esta condición depende de que en la primera se pueda realizar una jugada. Ahora se verificará que el jugador en turno tenga al menos una jugada válida en el tablero, en caso contrario se cambiará de jugador.
3. Una vez verificado que el jugador tiene al menos una jugada válida, se le solicita que cliquee un cuadro en el tablero, generando así, una coordenada que será evaluada para determinar si es una jugada valida, de no serlo imprime error y/o pasa a seleccionar otra.
4. Cuando se determina que una coordenada es válida, se procede a realizar la jugada, para esto el programa revisa en las 8 direcciones las líneas que cumplan con las normas de reversi para hacer una jugada. Inmediatamente después se descuenta una ficha, aumenta un turno, cambiar de jugador y calcular el número de casillas que poseen ambos jugadores en ese momento

Finalmente, por cada jugada realizada, se actualiza el dibujo del tablero con las posiciones y fichas cambiadas para ese turno. Al finalizar una partida completamente, imprime el jugador victorioso, o empate, de ser el caso y solicita al usuario una nueva partida o cerrar.

ESTADO ACTUAL

El programa tiene perfecta funcionalidad lógica, realiza las verificaciones y las jugadas correctamente, lee las coordenadas del mouse e imprime los resultados de los jugadores. Genera un tablero de interfaz, con el resultado de las casillas que tiene cada jugador y el turno cada jugador. En resumen, la parte lógica está a un 100% y la gráfica a un 75% de ser completada.

Es posible mejorar la interfaz, falta solicitar el nombre de los jugadores fuera de la ventada de comandos del sistema operativo.

El programa consta de 8 archivos indispensables para su funcionalidad. Para abrir el juego debe ejecutar el archivo Othello.py.

CONCLUSION

La implementación del algoritmo Othello obligó a pensar y buscar diversas maneras de cómo solucionar los problemas planteados, adquiriendo destreza en la resolución de errores de código Python y aprender a trabajar con varias herramientas del lenguaje no vistas en el curso de algoritmos y estructuras1. Aunado a eso, se comprende la importancia de trabajar en equipo y apoyarse en tu compañero.

El uso de la librería Pygame fue difícil y trabajoso, tal vez con unas pocas clases de algunos aspectos básicos ayudarían a que los alumnos no se sientas tan perdido con ella.