Pontificia Universidad
JAVERIANA
Bogotá

Proyecto: Juego Flow Free

Integrantes:
María Camila Aguirre Collante
Jessica Tatiana Naizaque Guevara

Departamento de Ingeniería de Sistemas

Docente:
Leonardo Florez Valencia

5 de octubre de 2022

Resumen

En este documento se presenta la documentación necesaria para conocer el funcionamiento del desarrollo del juego flow free desarrollado por las estudiantes. Dentro del informe, es posible encontrar el contexto, las indicaciones para los movimientos y demás descripciones relacionadas con el juego. **Palabras clave:** matriz, adyacencia, movimiento.

Índice

1. Introducción

Flow free es un juego presentado en un tablero con pares de puntos de colores que ocupan algunas posiciones del tablero. El objetivo del juego es conectar puntos del mismo color teniendo en cuenta que todo el tablero debe estar ocupado por algún color y cada par de colores debe estar conectado entre sí. El proyecto busca implementar el juego, por lo que en este documento se pueden encontrar distintas indicaciones del juego y de su implementación.

2. Implementación del juego

El desarrollo de la presente implementación se basa en el uso de un tablero o matriz $N \times N$ donde N es el nivel que ha elegido el jugador. Adicionalmente, durante la ejecución del programa, se considera un camino por cada uno de los colores existentes. En estos, se almacenan las casillas de dicho color en el orden en el que el usuario las ingresa, es decir, el camino que va llevando.

2.1. Tablero

El tablero de juego se define cuando el usuario selecciona la dificultad del nivel en el que va a competir. En este, se mostrará el número que representa cada casilla y algunas casillas con colores ya inicializados, estos últimos serán los extremos que deben unirse para completar el juego. En la figura 1, es posible observar una previsualización del tablero inicial para un nivel con dificultad 7×7.

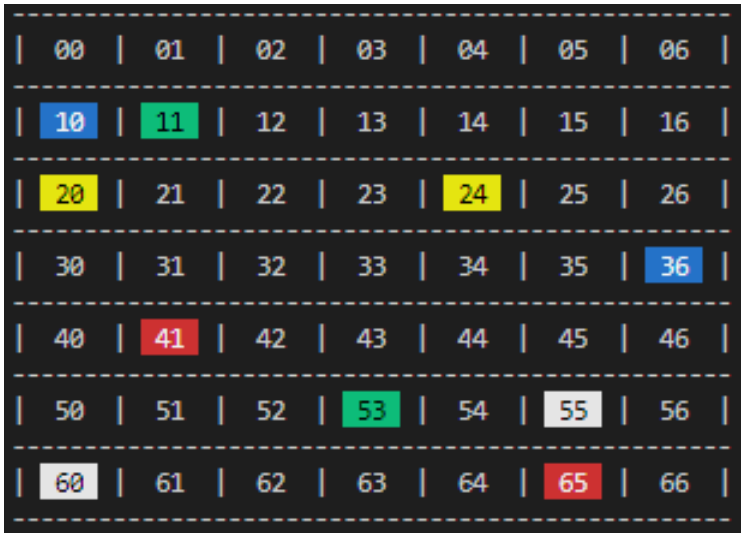

Figura 1: Tablero inicial para un juego aleatorio con dificultad 7×7

2.2. Niveles

Cuando se inicia el juego, se despliega un menú en el que el usuario debe elegir la dificultad del nivel que desea solucionar. Las diferentes modalidades desarrolladas para esta implementación son: 5×5, 6×6, 7×7, 8×8 y 9×9, por lo que el jugador deberá seleccionar una de estas opciones. Luego, el juego le mostrará una matriz aleatoria (entre 10 predefinidas) para la dificultad escogida y el usuario debe comenzar a elegir sus movimientos.

2.3. Movimientos

Para realizar un movimiento, el usuario debe ingresar dos parámetros separados por un espacio en blanco, el color y el número de la casilla con la que quiere jugar. Si se falla con los parámetros se encontrará con un *warning* indicando la razón por la que está erróneo, es posible encontrar estos fallos definidos en la sección 2.4. Para el color se debe ingresar la inicial del mismo, por lo cual, las equivalencias de color e inicial se pueden identificar en la sección 2.4.1. Para la casilla, se debe ingresar el número de la casilla a jugar, este número se puede encontrar en cada espacio del tablero. Un usuario no puede hacer cualquier movimiento en cualquier posición del tablero, sino que para que los parámetros del movimiento de entrada estén correctos, debe cumplir con ciertas condiciones, descritas en la sección 2.4.2. El juego acaba cuando todas las casillas del tablero tienen algún color en específico y las condiciones para ganar o perder se presentan en la sección 2.5

2.4. Parámetros incorrectos

Existen varios casos en los que los parámetros ingresados para el movimiento que quiere realizar el jugador sean incorrectos:

- Cantidad de parámetros: La entrada recibida en el programa se divide por espacios en blanco, lo que resulta en la cantidad de parámetros enviado. En el único caso en el que se acepta un solo parámetro es cuando el usuario ingrese 'X' o 'x', lo que indicaría que el jugador no quiere continuar en el nivel en el que se encuentra. Para los demás casos se espera que el usuario ingrese exactamente dos parámetros.

  Ejemplos incorrectos: 2 0 Y, B, $W33$, 05

- Tipo de dato de los parámetros: Luego de tener dos parámetros correctamente divididos, se verifica que ambos sean del tipo de dato esperado. Para el primero, la incial del color, debe ser una única letra (ya sea mayúscula o minúscula) que identifique al color. En caso de que el usuario ingrese el nombre del

color completo, el programa no identificará la acción que quiere realizar. Para el segundo, el número de la casilla, debe ser de tipo numérico, sin ningún tipo de caracteres especiales o letras adicionales.

Ejemplos incorrectos: *yellow* 23, *y* 2*i*, 33 *w*, *r*3*d* 42

- Número de la casilla: Inicialmente se verifica que la cantidad de dígitos sea exactamente dos y que cada uno de estos sea mayor o igual que cero y menor que el nivel seleccionado.

  Ejemplos incorrectos: *Y* 233, *B* 0, *W* 0000

- Sin casillas adyacentes del mismo color: Cuando el usuario ingresa un color y una casilla que quiere cambiar de color, esta nueva casilla debe estar adyacente a otra del mismo color, sin importar si es adyacente a un punto inicial o no.

2.4.1. Colores

La implementación del juego se diseñó teniendo en cuenta siete colores, se debe considerar que no es necesario que todos los tableros de los niveles tengan los siete colores. Estos colores y sus respectivas iniciales en inglés (para utilizarlas en los movimientos) son:

- **Amarillo:** *Y*
- **Azul oscuro:** *B*
- **Rojo:** *R*
- **Verde:** *G*
- **Blanco:** *W*
- **Azul cielo:** *C*
- **Morado:** *P*

2.4.2. Restricciones de los movimientos

Para que un jugador pueda tener un caso exitoso en la selección de un movimiento, debe tener en cuenta diferentes condiciones al momento de elegir la casilla y el color a jugar.

1. Cuando el usuario decide empezar por uno de los extremos de un color, debe continuar su camino por ahí, si selecciona como nueva casilla del color una adyacente al otro extremo, borra su camino previo y comienza el recorrido del color nuevamente.
2. El usuario puede agregar una casilla con un determinado color sí y solo sí esta nueva casilla tiene una adyacencia con alguna existente del mismo color.
3. El usuario puede seleccionar una casilla que ya tenga color, solamente si esta no es un extremo o una casilla inicial.
4. Cuando el usuario decide seleccionar una casilla que ya tiene un color y cumple con las condiciones determinadas, todo el camino del color anterior es eliminado por completo del tablero.
5. Si el usuario selecciona una casilla para ponerle x color y esta casilla ya tiene este color, el camino se elimina hasta esa casilla.

2.5. Fin del juego

El jugador debe completar estrictamente todas y cada una de las casillas del tablero para que el juego se complete, en caso de que falte al menos una casilla por color, el juego continuará solicitando los parámetros del turno.

Jugador pierde: Cuando el jugador completa todas las casillas del tablero con algún color y hay al menos un color que no conecte completamente sus extremos, el jugador ha terminado el juego pero de forma fallida. Es decir, el usuario no ha encontrado correctamente la solución.

Jugador gana: Cuando el jugador completa todas las casillas del tablero con algún color y absolutamente todos los colores tienen sus extremos conectados por medio de casillas adyacentes entre ellos. Es decir, el usuario encontró correctamente la solución.

2.6. Funcionamiento del juego

En la figura 2 se puede encontrar un diagrama de flujo que representa la interacción más importante del usuario con el juego, se muestra de manera general los movimientos que se realizan en cada tablero. Este diagrama se puede observar con mayor claridad en el Anexo 1 - Diagrama de flujo.

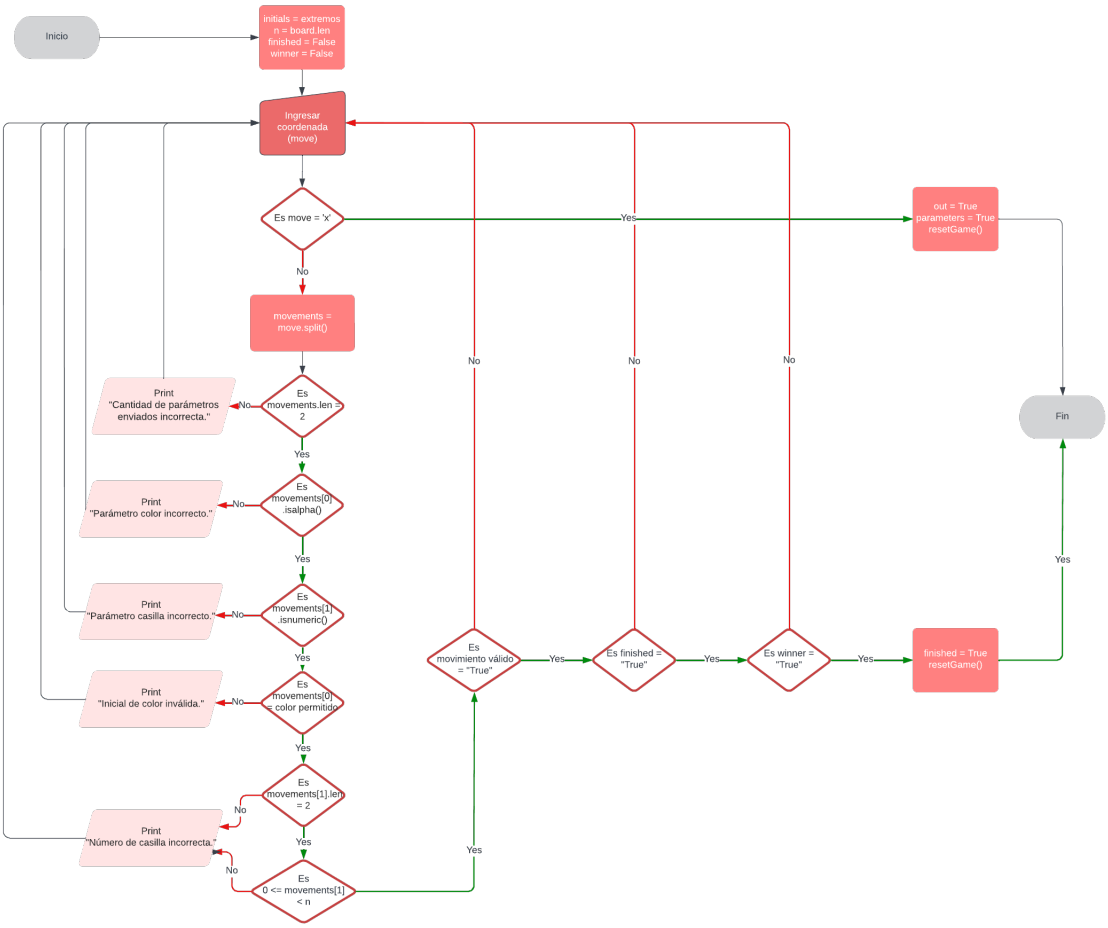

Figura 2: Diagrama de flujo “Movimientos Generales”

3. Documentación del código fuente

La implementación fue realizada en *Python*, se hizo uso de tres librerías principales: *copy*, *random* y *colorama*. La última es aquella que permite el cambio de colores en el fondo de un texto y en su letra, lo que facilita la impresión del tablero. A continuación, es posible encontrar los métodos desarrollados, sus descripciones y variables tanto de entrada como de salida. En los cuadros 4 a 18, es posible encontrar la documentación de cada método desarrollado para la implementación del juego "flow free".

defInitials	
Descripción	Método que llena cada lista de datos iniciales con las posiciones iniciales de cada color.
Entradas	*board (list)*: Tablero inicial del juego.
Salidas	Ninguna.

Cuadro 1: Contenido de la función *defInitials*

createBoard	
Descripción	Creación del tablero de juego a partir de la elección aleatoria del mismo.
Entradas	*dim (int)*: Dimensiones del tablero de juego.
Salidas	*board (list)*: Creación del tablero de juego.

Cuadro 2: Contenido de la función *createBoard*

showBoard	
Descripción	Muestra por consola el tablero de juego actual.
Entradas	*board (list)*: Tablero del juego.
Salidas	Ninguna.

Cuadro 3: Contenido de la función *showBoard*

checkE	
Descripción	Eliminar camino correspondiente a la casilla ingresada si esta se encuentra "llena".
Entradas	*element (string)*: Coordenada correspondiente a la casilla actual. *board (list)*: Tablero del juego.
Salidas	Ninguna.

Cuadro 4: Contenido de la función *checkE*

getAdj	
Descripción	Obtiene todas las casillas adyacentes que se encuentran del mismo color respecto a una casilla.
Entradas	*board (list)*: Tablero del juego. *color (string)*: Color correspondiente a la casilla para la cual se quieren hallar los adyacentes. *i (int)*: Fila correspondiente a la casilla para la cual se quieren hallar los adyacentes. *j (int)*: Columna correspondiente a la casilla para la cual se quieren hallar los adyacentes.
Salidas	*adjacents (list)*: Coordenadas de las casillas adyacentes que tienen el mismo color.

Cuadro 5: Contenido de la función *getAdj*

verifyPath	
Descripción	Verifica si existe otro camino del mismo color de la casilla ingresada.
Entradas	*color (string)*: Color correspondiente a la casilla actual. *board (list)*: Tablero actual de juego. *coordenate (string)*: Coordenada correspondiente a la casilla actual.
Salidas	'bool' *True*, cuando ya existe un camino desde algún extremo del color.

Cuadro 6: Contenido de la función *verifyPath*

addPath	
Descripción	Agrega casilla ingresada al camino correspondiente (haciendo las verificaciones necesarias).
Entradas	*color (string)*: Color correspondiente a la casilla actual. *board (list)*: Tablero actual de juego. *coordenate (string)*: Coordenada correspondiente a la casilla actual.
Salidas	Ninguna.

Cuadro 7: Contenido de la función *addPath*

removeE	
Descripción	Elimina coordenada del camino correspondiente.
Entradas	*color (string)*: Color correspondiente a la casilla actual. *element (string)*: Coordenada correspondiente a la casilla actual.
Salidas	Ninguna.

Cuadro 8: Contenido de la función *removeE*

removeNext	
Descripción	Respecto a una casilla, elimina todo el camino existente después de la misma.
Entradas	*color (string)*: Color correspondiente a la casilla actual. *coordenate (string)*: Coordenada correspondiente a la casilla actual. *board (list)*: Tablero actual de juego.
Salidas	Ninguna.

Cuadro 9: Contenido de la función *removeNext*

checkAdjacents	
Descripción	Verifica casillas adyacentes que serán eliminadas respecto a la coordenada recibida.
Entradas	*board (list)*: Tablero actual de juego. *coordBox (string)*: Coordenada correspondiente a la casilla ingresada.
Salidas	Ninguna.

Cuadro 10: Contenido de la función *checkAdjacents*

getIndex	
Descripción	Obtener la posición de una casilla en el camino correspondiente.
Entradas	*color (string)*: Color correspondiente a la casilla actual. *i (int)*: Fila de la casilla para la cual se quiere encontrar la posición. *j (int)*: Columna de la casilla para la cual se quiere encontrar la posición.
Salidas	*Retorna 'int'*: -1: Si la casilla no se encuentra en el camino. int: índice en el que se encuentra la casilla.

Cuadro 11: Contenido de la función *getIndex*

getPosPath	
Descripción	Obtener una coordenada a partir de una posición específica.
Entradas	*color (string)*: Color correspondiente al camino en el que se desea buscar. *pos (int)*: Posición (ínidice) de la cual se quiere obtener la coordenada.
Salidas	*Retorna 'string'* con la coordenada encontrada.

Cuadro 12: Contenido de la función *getPosPath*

checkBox	
Descripción	Verificar la casilla ingresada por el usuario se puede agregar e indicar la casilla que permite llegar a la misma.
Entradas	*board (list)*: Tablero actual de juego. *color (string)*: Color correspondiente a la casilla actual. *coordenate (string)*: Coordenada correspondiente a la casilla actual.
Salidas	*Retorna 'list':* lista[0] (string): Coordenada que permite llegar a la casilla actual. lista[1] (bool): Indica la existencia de una casilla adyacente que permita llegar a la casilla actual.

Cuadro 13: Contenido de la función *checkBox*

checkFinished	
Descripción	Indicar si el tablero ya fue llenado en su totalidad.
Entradas	*board (list)*: Tablero actual de juego.
Salidas	*Retorna 'bool'* que indica si el tablero de juego ya está completamente lleno.

Cuadro 14: Contenido de la función *checkFinished*

checkPath	
Descripción	Verificar si cada valor inicial (extremo) de un color tiene una casilla adyacente perteneciente al camino correspondiente.
Entradas	*initial (list)*: Extremos de un color. *path (list)*: Camino de un color. *dim (int)*: Dimensiones del tablero de juego.
Salidas	*Retorna 'bool'* que indica si el camino está completo.

Cuadro 15: Contenido de la función *checkPath*

checkWinner	
Descripción	Decidir si el jugador ya superó el nivel, verificando si el tablero está lleno de la manera correcta.
Entradas	*board (list)*: Tablero actual de juego.
Salidas	*Retorna 'bool'* que indica si el jugador completó el nivel.

Cuadro 16: Contenido de la función *checkWinner*

resetGame	
Descripción	Eliminar todos los datos almacenados en las coordenadas iniciales y en los caminos existentes.
Entradas	Ninguna.
Salidas	Ninguna.

Cuadro 17: Contenido de la función *resetGame*

selectMove	
Descripción	Controlar los movimientos del juego.
Entradas	*board (list)*: Tablero actual del juego.
Salidas	Ninguna.

Cuadro 18: Contenido de la función *selectMove*