Ceballos Vitale Pablo Guillermo, Gómez Tomás Alejandro

# Flick Color

Documento del Proyecto N°1 de la Materia Lógica para las Ciencias de la Computación

14/05/2018

# Contenido

## ASPECTOS GENERALES

### COMO JUGAR

#### FORMA DE LA INTERFAZ

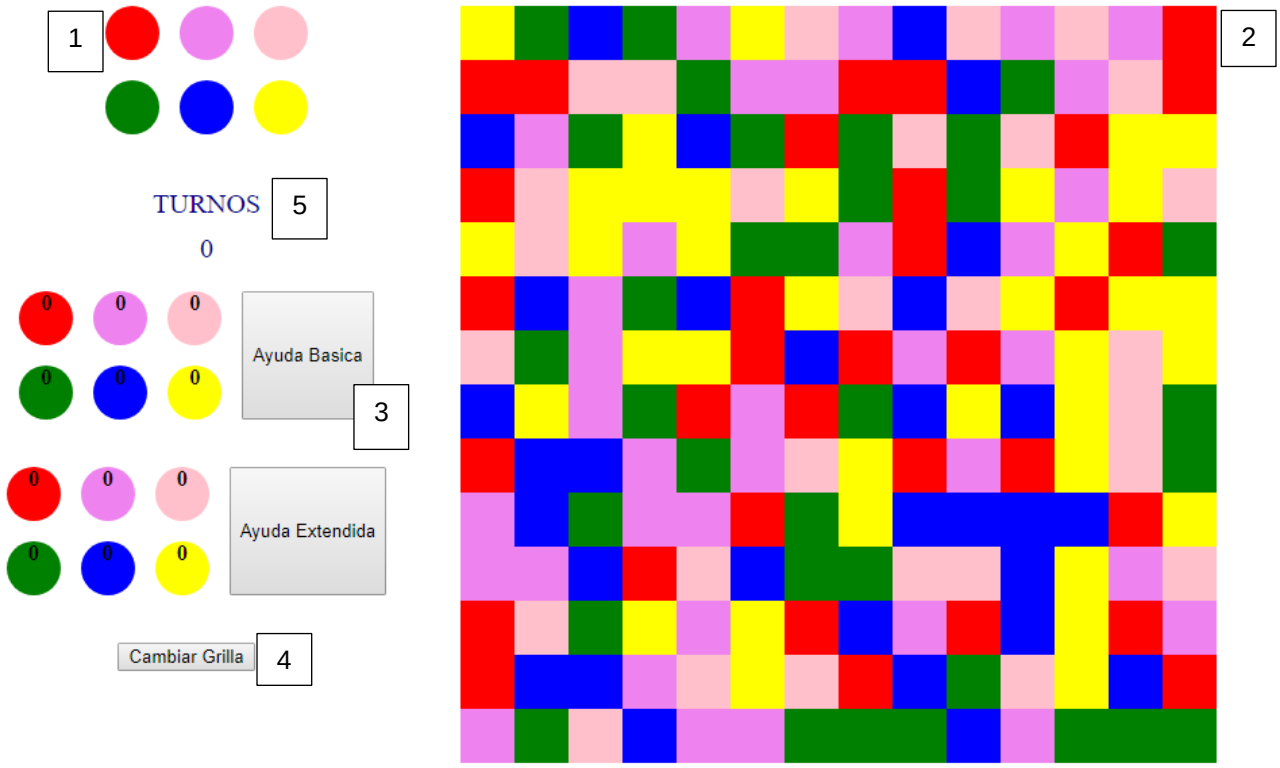


**Ilustración 1: Interfaz del Juego**

| | |
|---|---|
| 1 | Panel de botones para modificar el tablero de juego |
| 2 | Tablero de juego |
| 3 | Paneles de Ayuda |
| 4 | Botón de Cambio de Grilla |
| 5 | Número de Turnos |

## JUGABILIDAD

El objetivo del juego es pintar el tablero de un único color en la menor cantidad de turnos posible. Para esto, se disponen de 6 botones con los colores disponibles al jugador, como se podrá apreciar en la Ilustración 1 donde debe hacerse clic sobre cualquiera de los 6 colores para modificar el tablero de juego con el color deseado, cuando se presiona uno de los botones, la esquina superior izquierda del tablero cambiará al color deseado, y con eso cambiará a todas las casillas adyacentes a ella que sean del mismo color, hasta que se encuentren casillas adyacentes de colores distintos

Si el jugador todavía no ha empezado a jugar, puede optar por cambiar el tablero de juego actual, haciendo uso del Botón de Cambio de Grilla, esto hará que se cargue otro tablero del repertorio de tableros disponibles del juego

Si el jugador desea saber que jugada le conviene hacer para pintar la mayor cantidad de casillas, el jugador puede optar por pedir ayuda usando alguno de los dos botones de ayuda disponibles

- "Ayuda Básica" brindará información sobre la cantidad de casillas que serán pintadas para cada color en particular, luego de un turno
- "Ayuda Extendida" Brindará información sobre la mejor jugada posible que podría hacerse en dos turnos

## RESOLUCIÓN GENERAL DEL PROYECTO

Siendo un proyecto comprendido por distintas partes de diferente envergadura, asignar prioridades al proyecto fue crucial para resolverlo satisfactoriamente, siendo que resolver los problemas requeridos desde el lado del Lenguaje Prolog por el proyecto era necesario para asegurar el correcto funcionamiento del juego, decidimos que lo primero a resolver era encontrar la forma de que el código Prolog modifique el estado interno del tablero de juego correctamente, siendo que si se podía asegurar el correcto funcionamiento de un algoritmo que recorra el tablero y modifique el mismo, los demás requerimientos podían ser satisfechos fácilmente con mínimas modificaciones sobre el mismo algoritmo.

Luego, determinamos como segunda prioridad la implementación de los componentes JavaScript necesarias para albergar la lógica de las ayudas y del cambio de tablero de juego, fue necesario recurrir a fuentes de información adicionales, ya que nuestro conocimiento sobre el Lenguaje era limitado

Mientras se llevaba a cabo la implementación de las componentes gráficas, se desarrolló la parte lógica de las ayudas y del cambio de tablero de juego desde el lenguaje Prolog, los predicados usados para resolver estos problemas usan el algoritmo para recorrer el tablero del predicado que modifica el tablero en cierta manera que será descrita en detalle en sus secciones correspondientes.

## ESTRATEGIAS DE RESOLUCIÓN

### LADO DEL CLIENTE: LENGUAJE PROLOG

#### MODIFICAR EL TABLERO DE ACUERDO AL INPUT DEL USUARIO.

Se utiliza el predicado “flick”, llamado al presionar uno de los 6 botones en el panel de arriba. El predicado “flick” se encarga de llamar a “verificarColor”, el cual recorrerá toda la grilla de forma recursiva utilizando coordenadas X, Y, buscando la coincidencia de colores en celdas adyacentes a partir del color ubicado en la celda de arriba a la izquierda (coordenada 0,0) de la grilla; los ejes posicionados según la Ilustración 2

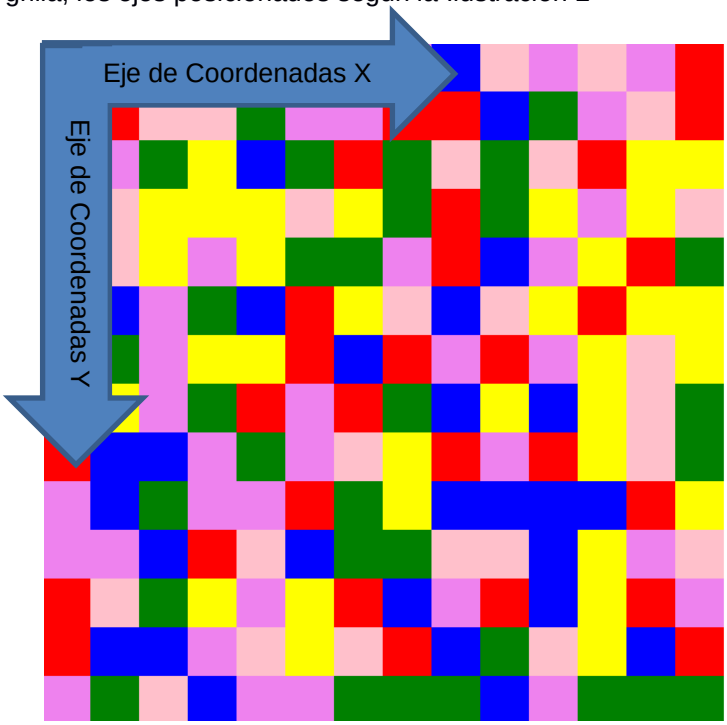


**Ilustración 2:Ejes coordenados**

Caso Base: Si al recorrer sobrepasé los límites de la grilla (las coordenadas son menores a 0 o mayores a 13), entonces dejo de buscar y dejo de modificar la grilla. En el caso de estar dentro de la grilla, si el color de la posición actual no coincide con el que estaba ubicado inicialmente en la posición 0,0, entonces me detengo y NO modifico la grilla.

Caso Recursivo: Si el color de la posición actual coincide con el que estaba inicialmente en la posición 0,0, entonces se cambia el color de la celda actual, al que vino en flick (el color que tenía el botón que presioné). Luego se repite el proceso para cada celda adyacente a la celda actual.

Cabe notar que la búsqueda y modificación de los colores de la grilla se realiza tal que, primero se recorren las distintas filas de la grilla (me muevo por el eje Y), y al llegar a la fila buscada, se recorren las columnas (recorro el eje X). Al llegar a la posición indicada, dependiendo de si el predicado usado fue “pintarCelda” o “buscarCelda”, o retorno el color que contiene la celda, o se modifica el color de la misma por otro.

**Lista Predicados Involucrados en la modificación del tablero de juego:**

flick(+Grid,+Color,-RGrid): Se encarga de llamar a “verificarColor”.

buscarCeldaVer/buscarCeldaHor(+X,+Y,+Grilla,-Res): Recorren la grilla primero por filas y después por columnas restando 1 a X e Y hasta llegar a 0. Luego se retorna el color en esa celda.

pintarCeldaVer/pintarCeldaHor(+X,+Y,+ColorNew,+Grilla,+NewGrilla): se comportan de manera idéntica al predicado buscarCelda, pero estas pintan la posición encontrada con el valor de la variable ColorNew (reemplaza el valor en la celda).

verificarColor(+X,+Y,+ColorActual,+ColorNew,+Grilla,-NewGrilla): Predicado que funciona de forma recursiva que se encarga de manejar la búsqueda y modificación del color de las celdas en la grilla.

**IMPLEMENTACIÓN DE LA AYUDA BÁSICA Y EXTENDIDA**

Se comienza con un predicado “cáscara” denominado “ayudaBasicaShell” que se encarga de llamar al predicado normal “ayudaBasica” 6 veces (1 vez para cada color) y retornar los resultados de cada uno.

Luego, en el predicado “ayudaBasica” se llama al predicado “flick” dos veces, la primera para pintar la grilla del color con el cual se pidió la ayuda, y la segunda para insertar 'n' en cada celda (esto no afecta a la grilla en pantalla). El propósito de 'n' es para mantener cuales son las celdas que se podrán pintar al presionar cierto color.

Luego se utiliza el predicado “contarCeldas” para pasar por cada celda de la grilla, y contar cuantas contienen 'n'. Hecho esto, se calcula cual es el color ubicado actualmente en la posición 0,0 de la grilla y se llama a “ayudaBasica” otra vez. Con este resultado, se restará a cada uno de los valores obtenidos previamente. Estos resultados serán los que se mostrarán en los marcos del panel de ayuda (al menos uno de esos valores será siempre 0). Estos valores representan cuantas celdas nuevas se agregarán después de la próxima pintada de la grilla.

**Lista Predicados Involucrados en la ayuda Basica:**

ayudaBasicaShell(+Grid,-ResR,-ResV,-ResP,-ResG,-ResB,-ResY): Llama a “ayudaBasica” 6 veces para obtener 6 resultados, luego lo llama otra vez con el color actual, y con ese resultado se lo resto a cada uno de los 6 valores obtenidos para obtener el cuantas celdas NUEVAS tendré luego de presionar cada uno de los botones para pintar la grilla. Retorna 6 valores enteros.

ayudaBasica(+Grid,+Color,-Res): Pinta dos veces a la grilla, y luego cuenta cuantas celdas tengo del mismo color comenzando desde 0,0.

calcularActual(+PosInicial,-ColorActual): Calcula cual de todos los 6 colores es igual a “PosInicial.”

contarCeldas(+X,+Y,+Color,+Grilla,-Res): Recorre la Grilla por completo, contando cuantas celdas con iguales a Color. Res es el resultado total de celdas iguales.

Por otra parte, se encuentra la “ayudaExtendida” que funciona de manera similar a la básica. Se comienza utilizando una “cáscara” denominada “ayudaExtendidaShell” tal que llama a “ayudaExtendida” 6 veces con cada uno de los colores.

Luego, en “ayudaExtendida”, se pinta la grilla del color actual, y se pasa a llamar a ayudaBasicaShell, la cual retorna 6 valores enteros (repitiendo el mismo proceso explicado arriba). Con esto luego busco el mayor valor de todos, el cual es el que voy a retornar y a ubicar en el lugar del color que corresponde.

**Lista Predicados Involucrados en la ayuda Extendida:**

ayudaExtendidaShell(+Grid,-ResR,-ResV,-ResP,-ResG,-ResB,-ResY): Llama a “ayudaExtendida” 6 veces, una para cada color.

ayudaExtendida(+Grid,+Color,-Res): modifica la grilla con el color provisto por la variable “Color”, luego llama a “ayudaBasicaShell” para obtener un “siguiente paso” de una manera similar. Una vez hecho esto, de los 6 valores provistos por “ayudaBasicaShell” me quedo solamente con el mayor. Luego se hace una llamada al predicado “ayudaBasica” con “Color” y le sumo ese valor al mayor que obtuve antes. Esto me resulta en cuantas celdas NUEVAS tendré en el mejor de los casos en 2 turnos.

ayudaAuxiliar(+Grid,+Color,-Res): Es un predicado usado para calcular lo que sería la ayuda básica pero sin llamar al shell junto con los 6 resultados que conlleva. Se encarga de calcular la cantidad de celdas que tiene la grilla de un color sin tocar nada y después de la primera simulación de la grilla pintada, se calcula la diferencia Res. Esto es usado para obtener el verdadero “ResBasico” usado por “ayudaExtendida”.

insertar(+X,+L,-LNew): Inserta X en una lista L y devuelve la nueva lista en la variable LNew.

calcularMayor(+L,+M,-Res): Calcula el mayor elemento de una lista L, comenzando por un M y devuelve dicho máximo en la variable Res.

## LADO DEL SERVIDOR: LENGUAJES JAVASCRIPT, HTML Y CSS

### DISEÑO Y FORMA DE LA INTERFAZ

Aunque una interfaz base fue provista por la cátedra, fue necesario extenderla para agregar las demás funcionalidades requeridas por el proyecto, dando lugar a la problemática de como diseñar la Interfaz Gráfica.

Decidimos que la mejor forma de organizarla era mantener todas las componentes interactivas sobre la barra lateral ya provista por la cátedra, para esto, definimos el resto de componentes que necesitaríamos en el Código HTML dentro de la sección correspondiente a esta barra lateral para que luego podamos acceder a ellas desde el Código JavaScript sin tener que definir las variables múltiples veces para proveerles de la Lógica necesaria para operar correctamente

La parte del Código CSS del proyecto solo fue modificada para proveer a las componentes agregadas a la interfaz de un aspecto gráfico adecuado

### COMUNICACIÓN CON PROLOG

El código JavaScript hace uso de una librería provista por la cátedra denominada “Pengines” para comunicarse con la lógica desarrollada en el lenguaje de Programación Prolog para hacer consultas a ciertos predicados implementados en dicho lenguaje, los mismos son:

- Flick(+Grid,+Color,-RGrid)
  - El cerebro lógico detrás del juego, este predicado se encarga de, dada la Grilla y un color seleccionado desde el lado del servidor, modificar la grilla de acuerdo a los requerimientos especificados en el proyecto y devolver una nueva grilla, que será levantada por el servidor para modificar el tablero gráfico del juego
- ayudaBasica(+Grid,+Color,-Res)
  - Se encarga de, dada la grilla, calcular para cada color posible cuantas celdas serán pintadas luego de un turno, la respuesta a cada color es levantada por el servidor para modificar los labels correspondientes
- ayudaExtendida(+Grid,+Color,-Res)
  - Se encarga de, dada la grilla, calcular para cada color cuantas celdas serán pintadas luego de dos turnos, calculando la máxima cantidad de celdas pintadas posibles para todas las combinaciones posibles de colores elegidos como segundo color

### IMPLEMENTACIÓN DEL CAMBIO DE GRILLA

A través del método “handleCambioGrilla” (el cual se activa al presionar el botón de cambio de grilla), es posible hacer ciclar la grilla por una de las 4 posibles. Simplemente se pide a través de pengines, que la grilla se coloree igual que la siguiente en el ciclo. Esto es desactivado al hacer la primera jugada en la grilla actual.

## CONCLUSIONES Y OBSERVACIONES GENERALES

El proyecto llevado a cabo resultó ser una experiencia altamente positiva debido a la experiencia obtenida en cuanto a implementación Web, un aspecto en el cual no contábamos con experiencia previa, adicionalmente, sirvió para afianzar los conocimientos sobre el Lenguaje de Programación en Lógica ya obtenidos durante el transcurso actual de la materia y aplicarlos en un entorno de desarrollo completamente nuevo para nosotros

OBSERVACIONES:

- Para el funcionamiento de las ayudas fue necesario hacer una llamada explícita con los 6 colores de la grilla en Prolog, tal que el predicado de cáscara llama al predicado normal 6 veces, una por color. Esto se realizó debido a que no logramos hacer funcionar las ayudas con un único predicado llamado 6 veces desde el código JavaScript a través de un ciclo que use el enumerado “colors”. Esta fue la forma en que logramos hacerlo funcionar.
- Se debe tener en cuenta que el programa puede ralentizarse a momentos en algunas ocasiones. No tenemos certeza si se debe a problemas de eficiencia en el código en sí, o por el retardo de la respuesta del internet.
- Puede ocurrir que si se deja el juego en “standby” por un rato entonces es posible que el programa deje de funcionar y sea necesario reiniciarlo. Desconocemos la causa por la cual ocurre esto