FLICKCOLOR

Informe Proyecto 2

Autores:

Cano,Micaela 114024

Moyano,Yohana Karina 105536

Materia: Lógica para Ciencias de la Computación

FlickColor :

¿Qué es FlickColor?

Consta de una grilla de 14 x 14, donde cada celda está pintada de uno de 6 posibles colores, y 6 botones, uno de cada color. Para jugar hay que presionar un botón de color C causa que la celda seleccionada actualmente de la grilla, pintada de color C', al igual que todas las del mismo color C' adyacentes a esta, y las de color C' adyacentes a éstas últimas, y así siguiendo, se pinten de color C.
Si C = C' entonces no se produce cambio alguno.

Objetivo del Juego

El objetivo del juego es lograr pintar todas las celdas de un mismo color, presionando los botones la menor cantidad de veces posible.

Requerimientos

Notación: usaremos adyacenteC* para referirnos a la clausura transitiva de la relación de
adyacencia entre dos celdas del mismo color. Es decir, diremos que una celda X es
adyacenteC* a una celda Y si ambas son del mismo color y además X es o bien adyacente a Y,
o existe una celda Z del mismo color que X e Y tal que X es adyacente a Z y recursivamente Z es adyacenteC* a Y.

Funcionalidad

En el juego presenta una grilla de 14x14 en la cual podremos elegir en qué casilla de esta queremos empezar a jugar. En el panel izquierdo hay una botonera con 6 colores para poder seleccionar.

El juego se desarrolla al ir seleccionando alguno de estos 6 colores sin olvidar que se aplicará la regla adyacenteC* para el nuevo color seleccionado.
El concepto de esta regla es que el nuevo color se esparcirá hacia todas las casillas del mismo color que se toquen entre sí.

Se mostrará en todo momento la cantidad de celdas que se encuentran capturadas actualmente, es decir, las celdas que son adyacenteC* a la celda de origen.

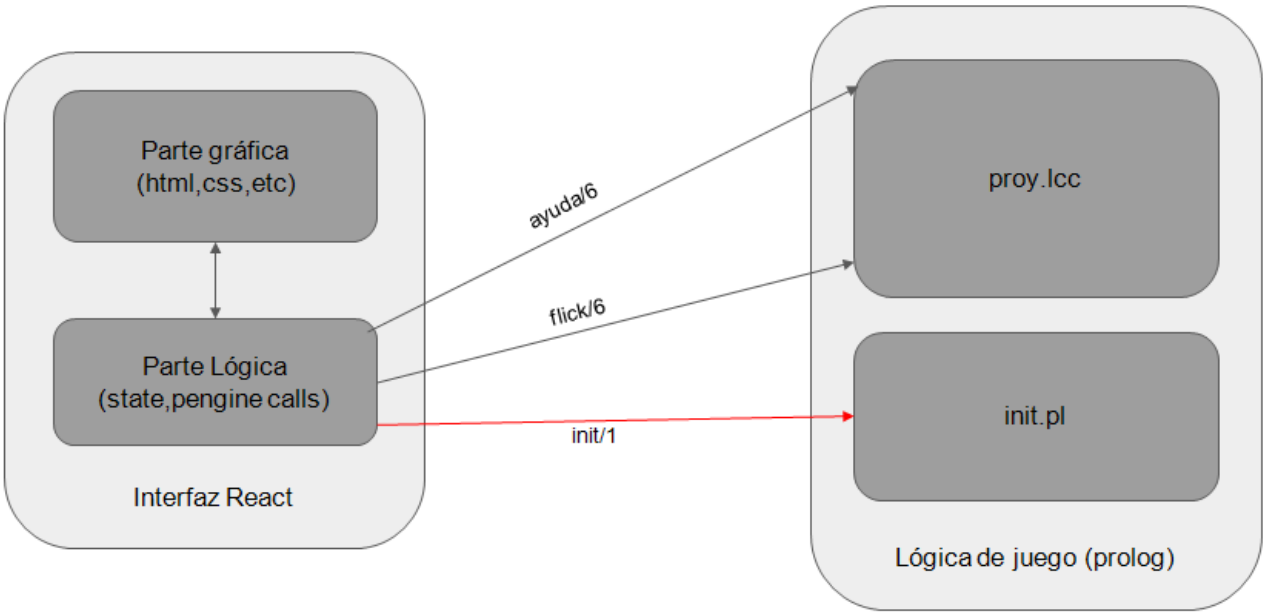
En el panel inferior se mostrará en todo momento, el historial de jugadas realizadas por el usuario (secuencia de colores presionados) desde el inicio de la partida.

En el panel derecho en todo momento estará disponible la ayuda “Strategy Help”. El usuario podrá ingresar en un input un valor del 1 al 5 (el cual refleja la profundidad para ser utilizada en la estrategia desarrollada en el apartado **prolog/Estrategia de Resolución**). Luego de presionar el botón “Help”, se mostrará un número con la mayor cantidad de celdas que pueden ser capturadas y la lista de colores que se deben seleccionar para obtener ese valor.

El juego terminará cuando el tablero quede completado de un solo color.

Arquitectura de FlickColor :

Esquema de la arquitectura (simplificada)



La primera llamada realizada de parte de react a prolog es la de **init/1** para generar el tablero.

La segunda llamada realizada es **flick/6**, la cual devolverá la cantidad de capturados y la Grilla actualizada.

La tercera llamada realizada es **ayuda/6** la cual es opcional. Si se utiliza devolverá un número con la mayor cantidad de celdas que pueden ser capturadas y la lista de colores que se deben seleccionar para obtener ese valor.

React :

Podemos encontrar 3 clases principales que extienden de React Component: Game, Board y Square.

En el constructor de la clase Game encontramos:

```
this.state = {
     turns: 0,
     grid: null,
     complete: false,  // true if game is complete, false otherwise
     waiting: false,
     origin: undefined,
     clicks: [],
     capturadas: 0,
     pe:'',
     StrategyBest: 0,
     listColors: [],
     playing: false,
   };
```

Con estos datos, inicializamos el estado de la grilla.

```
   this.handleClick = this.handleClick.bind(this);
   this.handlePengineCreate = this.handlePengineCreate.bind(this);
   this.onOriginSelected = this.onOriginSelected.bind(this);
   this.pengine = new PengineClient(this.handlePengineCreate);
   this.handleHelp = this.handleHelp.bind(this);
```

Y enlazamos estos manejadores de eventos a una instancia.

Llamadas a Prolog desde React

flick/6

Desde React en la clase Game llamamos al predicado flick/6 de prolog con los siguientes parámetros:

```
"flick(" + gridS + "," + color + ", Grid, " + fila + "," + col + ", Capturadas)";
```

Este me devuelve:

- **Grid**: Una nueva grilla con los movimientos actualizados que vamos realizando.
- **Capturadas:** Cantidad de elementos adyacentes que están capturados actualmente.

ayuda/6

Desde React en la clase Game llamamos al predicado ayuda/6 de prolog con los siguientes parámetros:

```
"ayuda(" + gridA + "," + fila + "," + col + "," + PE + ", Best, List)";
```

Este me devuelve:

- **Best**: Número de celdas capturas que obtiene la secuencia de (a lo sumo) PE movidas (colores).
- **List:** Secuencia de (a lo sumo) PE movidas (colores), que consiguen capturar la mayor cantidad de celdas.

Prolog :

Estrategia de Resolución:

La estrategia adoptada en el predicado **flick**, va detectando y cambiando de color los adyacentesC* en el mismo recorrido. Al ir cambiando de color se está evitando ya visitar repetidos.

En la estrategia adoptada en el predicado **ayuda**, retorna un número con la mayor cantidad de celdas que pueden ser capturadas y la lista de colores que se deben seleccionar para obtener ese valor. En la estrategia utilizada, recorremos un árbol por cada color que pertenece a la grilla, excepto el color de la celda actual. Y luego con ayuda de otro predicado buscamos cuál de esos recorridos es el que posee mayor cantidad de celdas capturadas y retornamos ese valor.

En la estrategia adoptada en el predicado **adyCStar** utilizamos el código implementado por la cátedra, el cual devolverá la cantidad de celdas del mismo color y un arreglo de celdas visitadas.

Predicados principales utilizados:

- **flick/6:** Dada una Grilla, color Nuevo, posición X, posición Y, devuelve la cantidad de capturados actuales y la Grilla con los colores actualizados.
- **obtenerElemGrilla/4:** Dada una posición X, posición Y, una grilla, obtengo el elemento E de la grilla en la posición (X,Y).
- **cambiarC/6:** Dada una Grilla, color actual, color nuevo, una posición X, posición Y, devuelve una Grilla con los colores actualizados.

- **replace/4:** Dada una posición X, una lista y un elemento nuevo, devuelve una lista con el elemento nuevo reemplazado en la posición (X).
- **cambiarElemG/5:** Dada una posicion X, posicion Y, una grilla y un color nuevo, devuelve una nueva grilla con el elemento que se encuentra en la posicion (X,Y) cambiado de color.
- **adyacentes/6:** Dada una grilla, color actual, color nuevo,y dos posiciones devuelve una grilla actualizada.
- **allColorsButC/2:** Dada un color C y una lista de colores, devuelve la lista de colores sin el color C.
- **ayuda/6:** Dada una Grilla, una posición X, una posición Y, una profundidad PE, devuelve el mejor número de celdas capturadas y una secuencia de colores de profundidad PE.
- **helpAll/7:** Dada una Grilla, una lista de colores, una posición X, una posición Y, una profundidad PE, devuelve el mejor número de celdas capturadas y una secuencia de colores de profundidad PE.
- **helpOne/7:** Dada una Grilla, un color, una posición X, una posición Y, una profundidad PE, devuelve el mejor número de celdas capturadas y el Color asociada a ese recorrido.
- **mayorC/6:** Dada la mayor cantidad de celdas capturas en helpOne, la mayor cantidad de celdas capturas en el helpAll, la secuencia obtenida en helpOne, la secuencia obtenida en helpAll, devuelve el mejor valor de cantidad de celdas capturadas y la mejor secuencia de listas.
- **adyCstar/4:** Dada una posición Origen, una Grilla, devuelve una lista de visitados y la longitud de la lista visitados.
- **adyCStarSpread/4:** Dada un lista de celdas pendientes, una lista de celdas visitadas y una Grilla, devuelve una lista de celdas visitadas.

- **adyC/3:** Dada un elemento P, una Grilla, devuelvo los adyacentes del elemento P.
- **ady/3:** Dada un lista de posición (X,Y), una Grilla, devuelve una lista con una posición de los adyacentes.
- **color/3:** Dada una posición (X,Y) y una Grilla, devuelve el elemento de la posición.

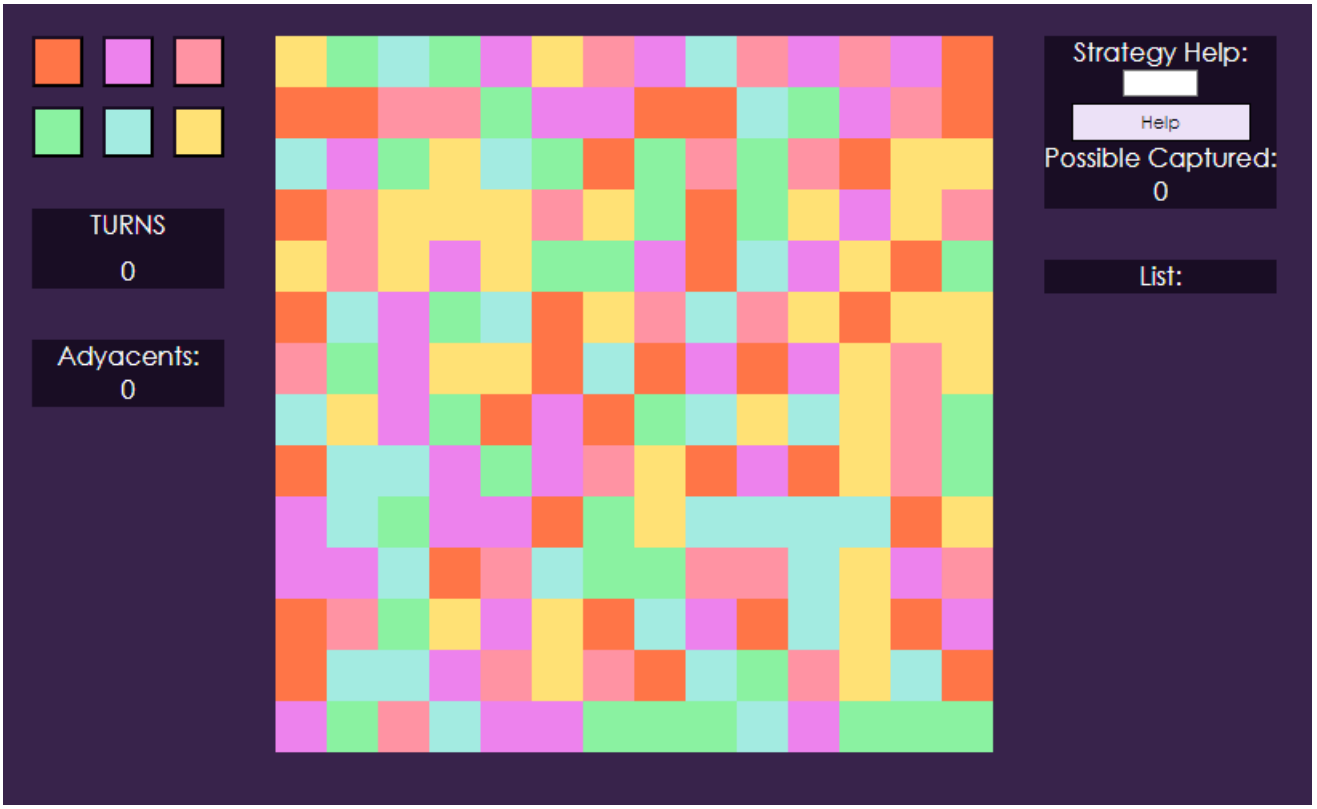
Predicados predefinidos:

Los predicados predefinidos que utilizamos son los siguientes:

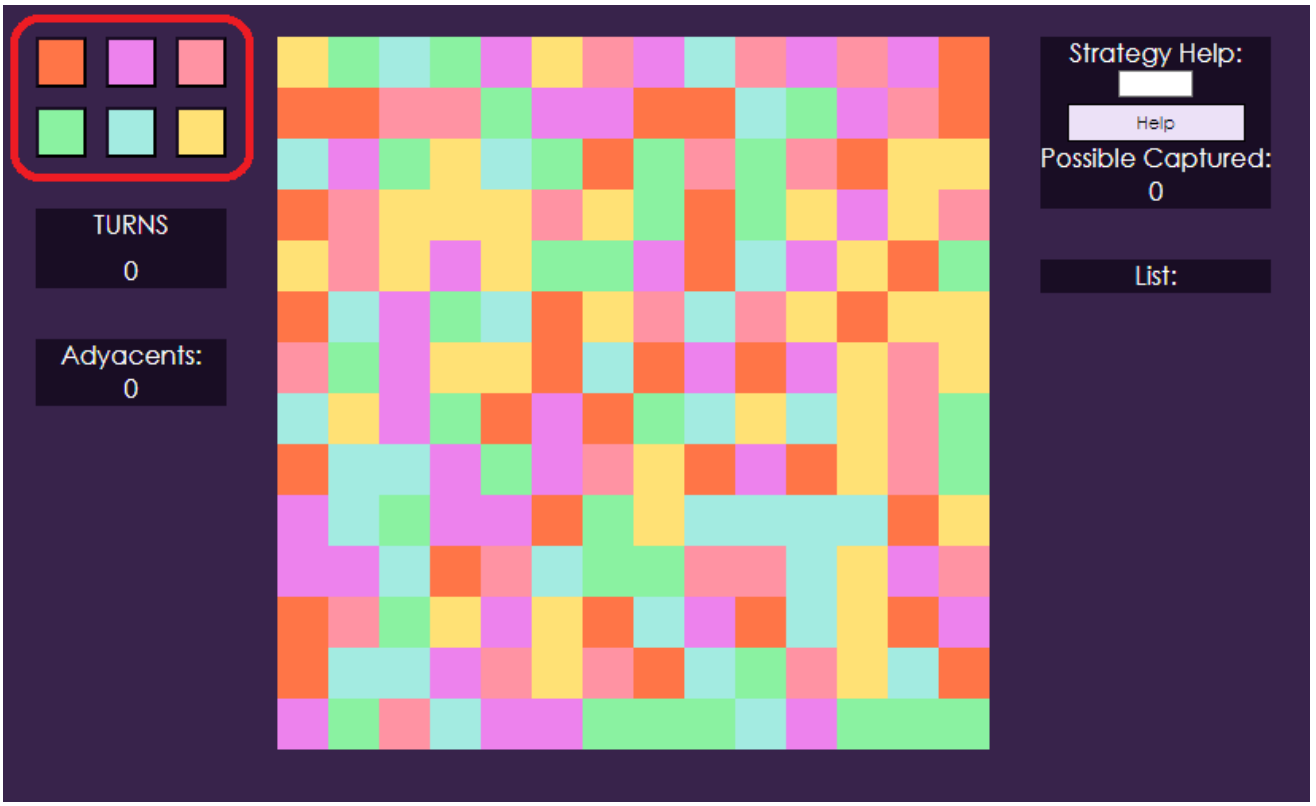
- **nth0/3:** Buscamos un elemento dada una posición.
- **nth0/4:** Reemplazamos un elemento dada una posición.
- **delete/3:** Elimina un elemento de una lista.
- **length/2:** Devuelve la longitud de una lista.
- **findall/3:** Devuelve una lista con todas las soluciones posibles que cumplen con la condición dada.
- **not/1:** Verdadero si no se puede la condición dada.
- **member/2:** Verdadero si el elemento es miembro de la lista.
- **append/3:** Dada una lista L1 y una lista L2, concatena L1 con L2.

Manual de Usuario :

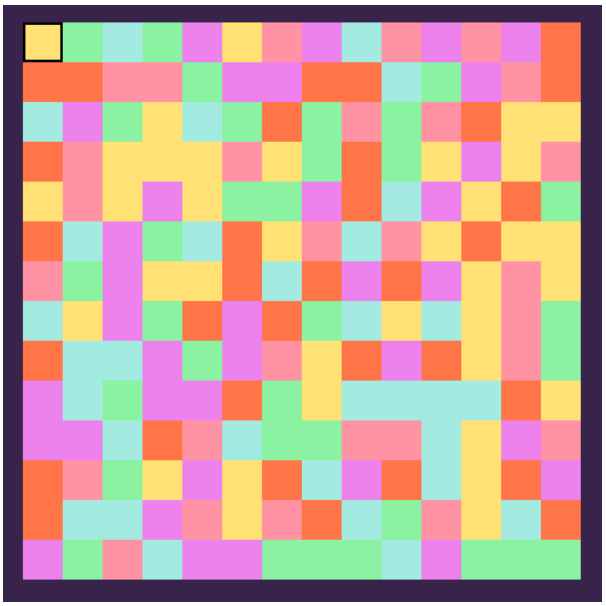
Al comenzar el juego tendremos una grilla sin resolver.

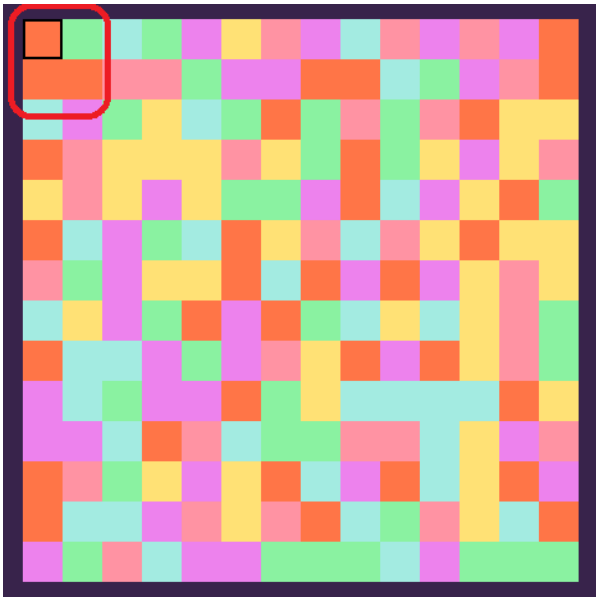


En el panel izquierdo se muestran 6 botones con colores distintos para seleccionar.



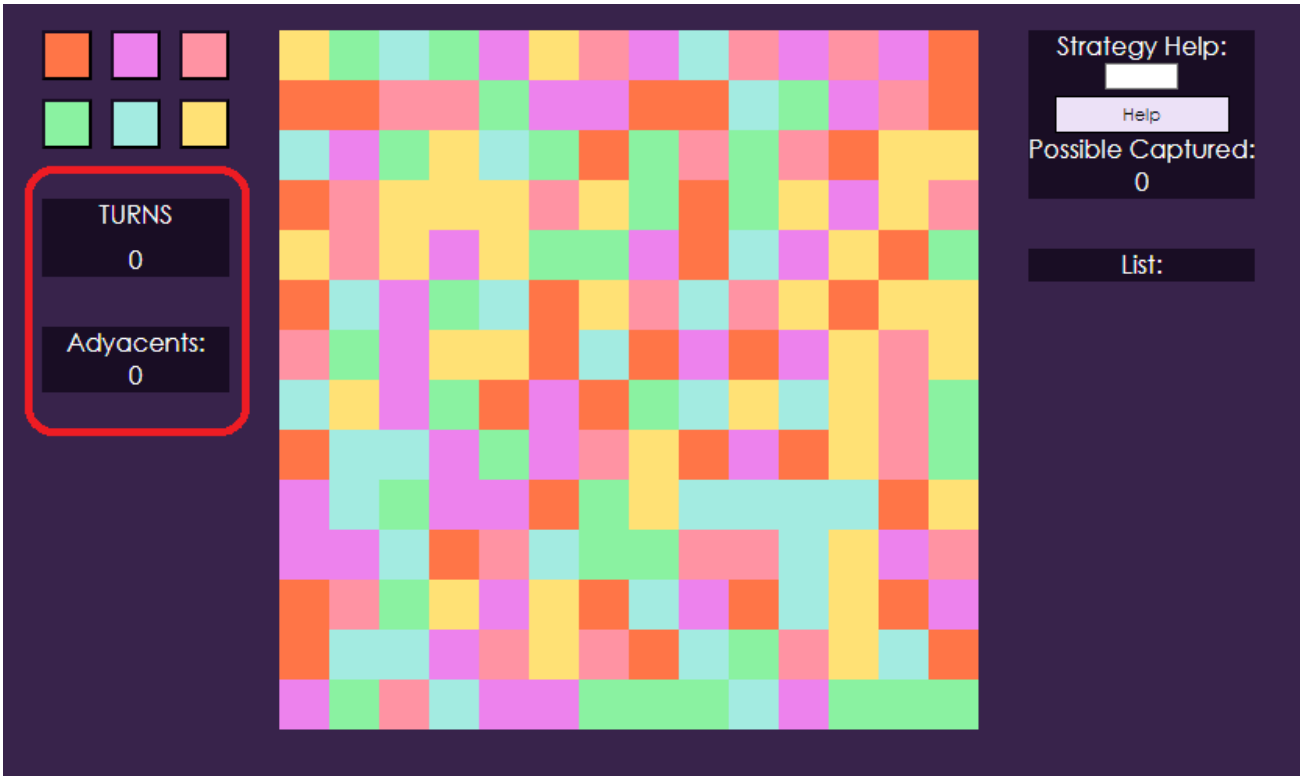
El juego inicia al hacer clic en una celda, si esto no ocurre se iniciará con la celda superior izquierda.



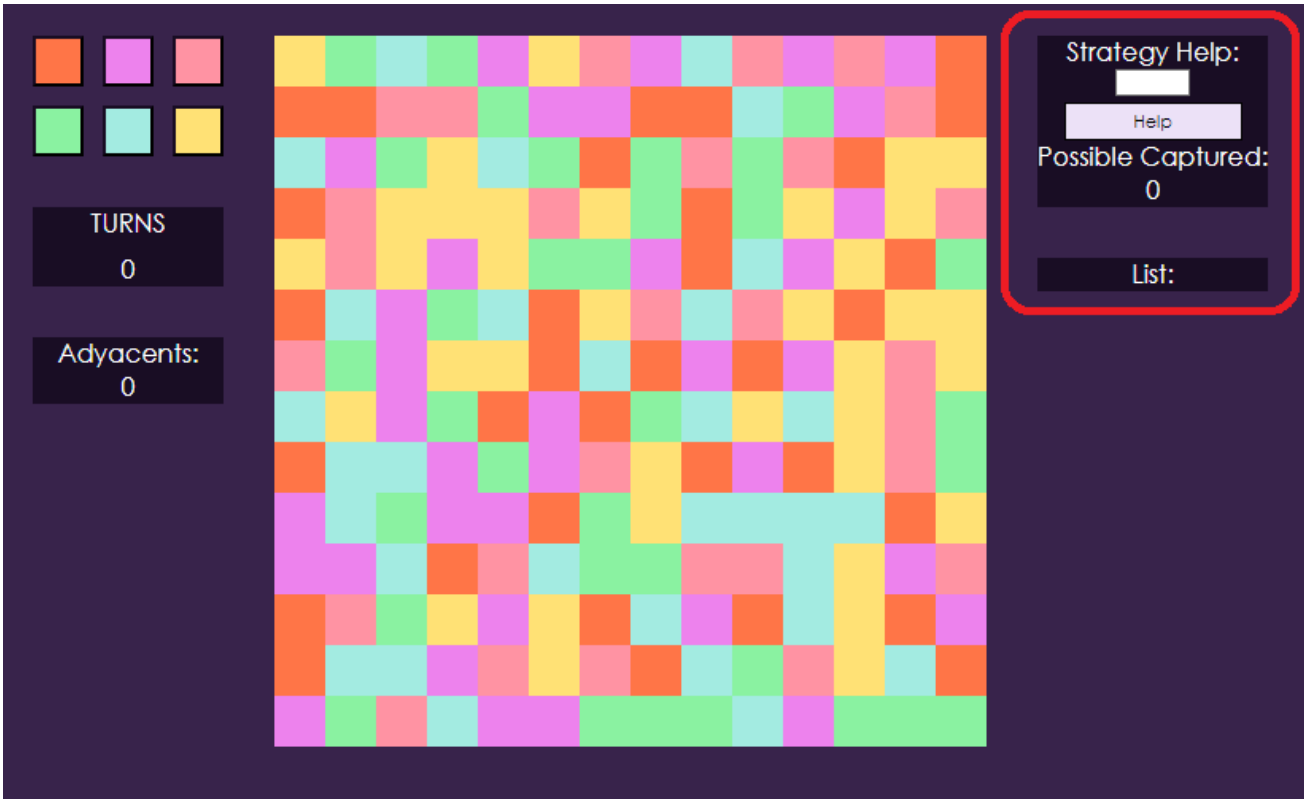


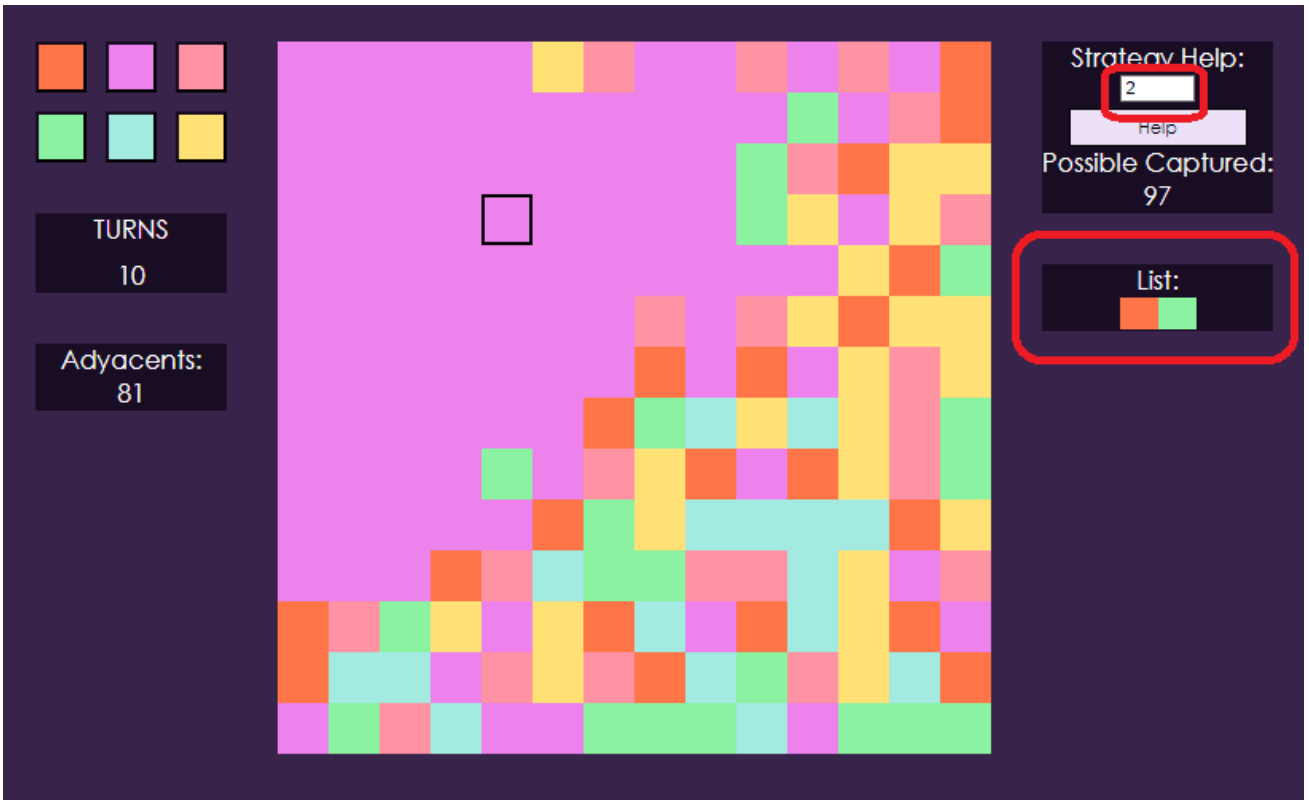
Al elegir una celda o utilizar la celda por defecto se podrá visualizar que sus adyacentes del mismo color se van reemplazando, al no haber más adyacentes para pintar se espera un nuevo clic.

En el panel izquierdo se podrá visualizar en todo momento la cantidad de jugadas realizadas y la cantidad de adyacentes capturados.

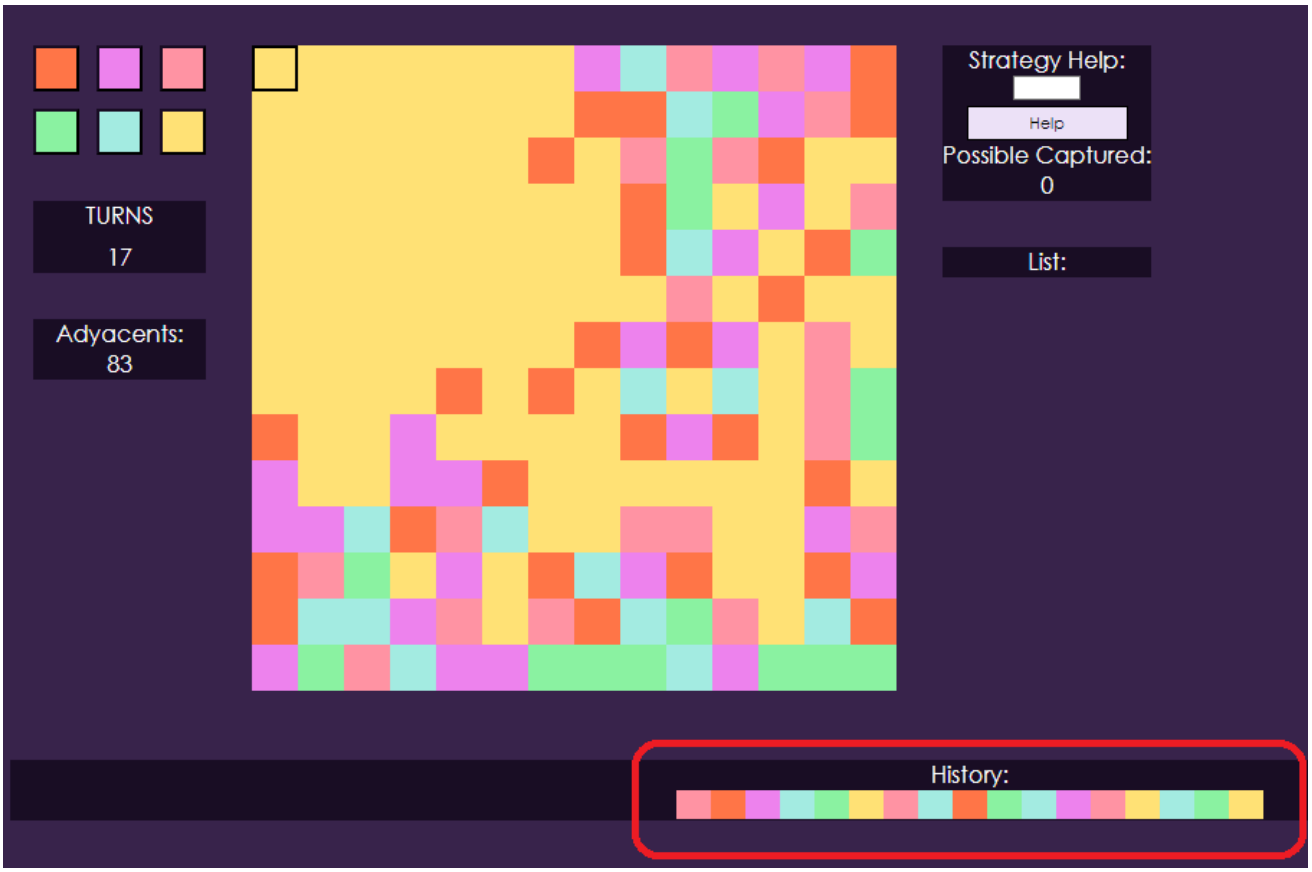


En el panel derecho se podrá visualizar en todo momento el botón de ayuda que nos permitirá ingresar un valor del 1 al 5 para saber la mejor jugada en la cantidad de pasos ingresada y también se podrá visualizar la cantidad de adyacentes capturados por esta solución.





Tendremos un historial en la parte inferior para visualizar las secuencias de colores presionados.



El juego finaliza cuando la grilla queda completa de un solo color y la cantidad de adyacentes sea 196.