Informe Proyecto 1

Valentino Villar - Alejo Quintana

Comisión Nº15

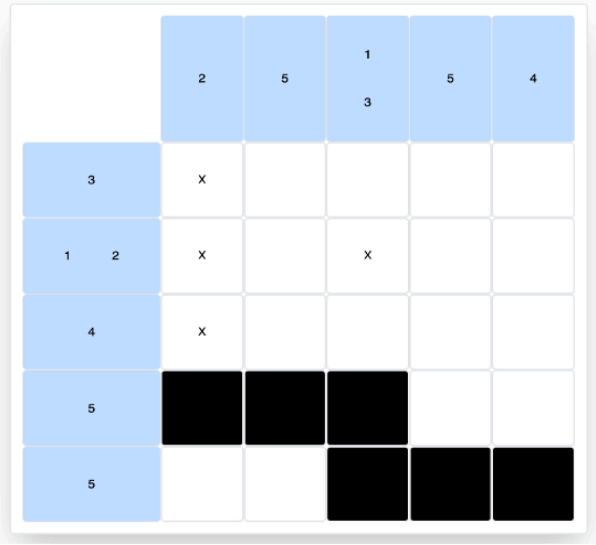
Grilla

La grilla interactiva que se le presenta al usuario es una lista de listas en Prolog, con algunas casillas ya pintadas, otras con una X a forma de una ayuda inicial para poder resolver el nivel y el resto de las casillas vacías. Esta grilla se encuentra en el archivo init.pl y a continuacion se muestra como se ve la grilla en el código y como se ve luego en la ejecución del programa:

```
init(
[[3], [1,2], [4], [5], [5]],    % RowsClues

[[2], [5], [1,3], [5], [4]],    % ColsClues

[["X", _ , _ , _ , _ ],
 ["X", _ ,"X", _ , _ ],
 ["X", _ , _ , _ , _ ],      % Grid
 ["#","#","#", _ , _ ],
 [ _ , _ ,"#","#","#"]
]
).
```



Método Put

Para que las celdas se pinten/despinten se utiliza el método Put el cual es llamado cada vez que el usuario clickea la grilla.

Además de insertar el contenido dependiendo del modo en el que se encuentre el juego, en el put se agregó el chequeo que se encarga de decirnos si fue satisfecha la fila o columna.
Utilizamos copy_term() para copiar la grilla y la fila a chequear para evitar problemas con el backtracking.
Luego llamamos al método clue() que se encarga de devolvernos un elemento en cierta posición en un arreglo, que básicamente nos devuelve la clue respectiva de cada fila o columna.
A continuación calculamos la traspuesta de la grilla para obtener la columna de la posición dada.
Finalmente buscamos la validación de las clues tanto en la fila como en la columna con el método search_clues().

```
put(Content, [RowN, ColN], RowsClues, ColsClues, Grid, NewGrid, RowSat, ColSat):-
    replace(Row, RowN, NewRow, Grid, NewGrid),
    (replace(Cell, ColN, _, Row, NewRow),
    Cell == Content
        ;
    replace(_Cell, ColN, Content, Row, NewRow)),
    copy_term(NewGrid, ClonedGrid),
    copy_term(NewRow, ClonedRow),
    clue(RowN, RowsClues, RowClue),
    clue(ColN, ColsClues, ColClue),
    col_to_row(ColN, ClonedGrid, Column),
    search_clues(RowClue, ClonedRow, RowSat),
    search_clues(ColClue, Column, ColSat).
```

Switch para cambio de modo

Para seleccionar el modo en el que el usuario quiere estar (si quiere colocar una X o pintar la celda) se le brinda un switch que le permite cambiar de un modo a otro. Según el modo en el que se encuentre el Switch se decidirá si se coloca una X o si se pinta la celda. A nivel código, lo que hace el switch es modificar el parámetro content que se le envía al método put, cambiándolo constantemente entre “X” o “#”, ya que las celdas pintadas se identifican con un “#”. En caso de que en la celda clickeada ya haya un símbolo del mismo tipo este es eliminado, y

en caso de ser de otro tipo, es reemplazado. A continuación se muestran los dos posibles estados del Switch y el código asociado que modela el comportamiento deseado:

NONOGRAM. NONOGRAM.


```
const content = painting ? '#' : 'X'; // Content to put in the clicked square.
```

```
<SwitchBtn painting={painting} onClick={() => setPainting(!painting)}/>
```

Verificación de pistas

A medida que avanzamos en el Nonograma, y vamos pintando celdas eventualmente iremos completando las pistas que se encuentran a los lados del tablero. En caso de que se complete alguna pista, su recuadro azul en el que se encuentra se convertirá en un recuadro verde.
El algoritmo que se encarga de chequear si se cumplieron las clues lo pensamos de la siguiente manera:
Dado una clue de una fila o columna (ej. [1,2]) y un array con elementos (ej. [X,#,X,#,#]), definimos un método cascara para que recorra el arreglo hasta el primer “#”, luego de encontrarlo llamamos al método check_clues() para que a partir de ahí empiece a recorrer validando la clue. Cuando encuentra un “#” se le resta uno a la clue y se vuelve a llamar al método. Una breve traza:
check_clues([1,2], [#,X,#,#], Valid).

check_clues([0,2], [X,#,#], Valid).
Se sigue recorriendo mientras el arreglo de clubes tenga elementos y el arreglo que podría ser tanto una columna o una fila, esté vacío.

Uno de los inconvenientes que tuvimos apareció cuando queríamos discriminar el "_" debido a que usábamos "=" para comparar, esto nos ocasiona que por ejemplo _ = "X" sea siempre true. Esto fue solucionado utilizando "==".
Otro inconveniente que surgió, fue cómo diferenciamos el caso en el que se trataba de una clue con un salto en el medio, por ejemplo [1, 1] o una clue sin salto como puede ser [2]. Por ejemplo usando el siguiente arreglo [#,X,X,X,#]:

- La clue es [1, 1], encuentro la primer coincidencia y ahora la clue es [1] y el arreglo que me queda por recorrer es [X,X,#]
- La clue es [2], encuentro la primer coincidencia y ahora la clue es [1] y el arreglo que me queda por recorrer es [X,X,X,#]

Se trata de dos situaciones en las que me encuentro parado casi en el mismo escenario pero tengo que lograr que en un caso me de falso y en otro true, ya que la clue [1, 1] se verifica pero la [2] no. Para solucionar esto, incorporamos un caso a nuestro método check_Clues el cual en el caso de las clues sin salto resta uno al numero de la clue pero además controla que el siguiente elemento del arreglo sea un #, caso contrario, retorna false.

Victoria

A medida que el jugador vaya avanzando y completando clues estas se irán coloreando de verde mientras que dentro del código se agregan a un arreglo llamado RowSat o ColSat dependiendo si es una pista para fila o para columna. Esos arreglos almacenarán

las clues que se encuentran resueltas hasta el momento. Una vez que RowSat y ColSat tienen todas las clues del tablero dentro, es decir, están todas las pistas resueltas en simultáneo, el jugador ha ganado. Para poder mostrar eso, se compara el tamaño de RowSat y ColSat con RowsClues y ColClues. En caso de que sean iguales, significa que todas las clues han sido resueltas. Se envía una alerta al jugador de que el nivel fue completado.

Test

Algunos de las grillas/niveles que se utilizaron para comprobar el correcto funcionamiento del sistema fueron:

Caso de prueba de 10x10 --- Signo de Pregunta

```
init(
[[6],[2,2],[2,2],[2],[3],[4],[2],[0],[2],[2]], % RowsClues
[[0],[2],[3],[1],[1,2,2],[1,2,2],[1,2],[6],[4],[0]], % ColsClues
[[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ], % Grid
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ]).
```

Caso de prueba de 10x10 --- Coco

```
init(
[[4], [6], [3,4], [2,5], [4,5], [10], [10], [8], [6], [4]],% RowsClues
[[4], [6], [2,7], [10], [2,1,5], [10], [10], [8], [6], [4]], %
ColsClues
[[ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ], % Grid
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
```

```
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
 [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ]]).
```

Caso de prueba 10X10 --- Pulpo

```
init(
[[2], [4], [6], [1,6,1], [1,6,1], [2,4,2], [10], [8], [1,2,2,1],
[3,3]], %RowClues
[[4,2], [3,1], [3,4], [8], [8], [8], [8], [3,4], [3,1], [4,2]],
% ColClues

    [[ _ , _ , _ , _ ,"#","#", _ , _ , _ , _ ],
     [ _ , _ , _ ,"#", _ , _ ,"#", _ , _ , _ ],
     [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],        % Grid
     ["#", _ , _ , _ ,"#","#", _ , _ , _ , _ ],
     ["#", _ , _ , _ , _ , _ , _ , _ , _ , _ ],
     [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
     [ _ , _ , _ , _ , _ , _ , _ , _ , _ ,"#"],
     [ _ , _ , _ , _ , _ , _ , _ , _ , _ , _ ],
     [ _ , _ ,"#","#", _ , _ , _ , _ , _ ,"#"],
     [ _ , _ , _ , _ , _ , _ , _ , _ , _ ,"#"]
    ]
```

Extras

Como extra podemos mencionar la utilización del Framework Tailwind CSS que nos permite hacer un estilaje mucho más sencillo y rápido.

Al final del juego cuando el jugador gana, se le emite en pantalla una alerta indicando que ganó con un botón que le permite reiniciar el nivel y comenzar de nuevo.