

Tetris en Haskell

Jinxiang Ye - d21M081

José Márquez Carqués d21M047

Inés Colmenero López - d21M066

Índice

Índice.....	2
1. Introducción.....	3
2. Librerías.....	4
3. Código.....	6
3.1. Bloque físico.....	6
3.2. Bloque lógico.....	8
4. Aprendizaje.....	10
5. Conclusiones.....	11
6. Bibliografía.....	12

1. Introducción

Se documenta el desarrollo de una versión del conocido juego Tetris, utilizando el lenguaje de programación Haskell. El Tetris destaca por su simplicidad y su alto nivel de adicción, lo que lo convierte en un proyecto ideal para explorar diversas técnicas de programación y conceptos algorítmicos. Implementar Tetris en Haskell ha presentado un desafío único debido a la naturaleza funcional y declarativa del lenguaje.

El enfoque de este proyecto es, por un lado, proporcionar una comprensión profunda del uso de la programación funcional en la creación de un videojuego, y por otro lado, demostrar las capacidades de Haskell en la gestión de estados, la manipulación de estructuras de datos y la creación de interfaces gráficas.

En primer lugar, comenzaremos hablando de la instalación y compilación de las librerías utilizadas, siguiendo con unos comentarios sobre el código que incluye interfaz y cómo se muestran y se mueven las piezas. Finalizando con el aprendizaje y los resultados obtenidos.

2. Librerías

En cuanto a la instalación de dichas librerías, seguimos una serie de pasos. En primer lugar, añadimos las variables de entorno que más tarde nos servirán para buscar las librerías instaladas:

```
LD_LIBRARY_PATH = C:\msys64\mingw64\bin
PKG_CONFIG_PATH = C:\msys64\mingw64\lib\pkgconfig;
                  C:\msys64\mingw64\share\pkgconfig
```

También será necesario añadir las direcciones `C:\msys64\usr\bin` y `C:\msys64\mingw64\bin` a la variable de entorno `Path` del sistema.

Luego, ejecutamos en el shell de Haskell los siguientes comandos, los dos primeros para instalar gtk y el último para comprobar si la librería se ha instalado correctamente.

```
pacman -Syu
pacman -S mingw-w64-x86_64-gtk2
pkg-config --modversion gtk+-2.0
```

Para usarlo, creamos un entorno del juego usando cabal con el comando:

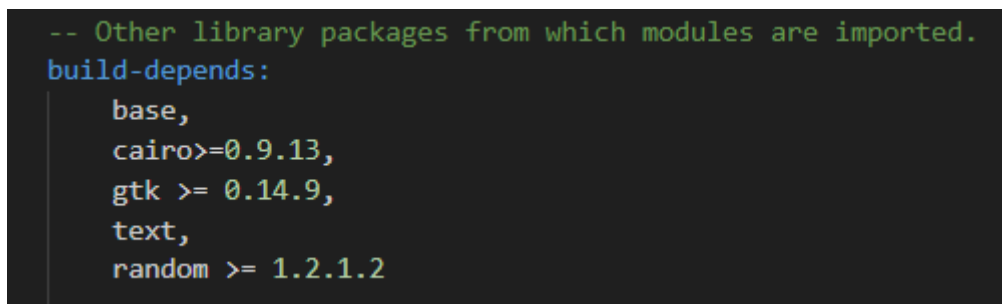
```
cabal init mytetris --non-interactive
```

Con ello, se crea una carpeta `mytetris` con el fichero principal del juego (`app/Main.hs`) y `mytetris.cabal` que hace posible la compilación. En este entorno son útiles los comandos:

```
cabal build
cabal run
```

Donde el primero de ellos compila el juego y el segundo lo ejecuta.

Finalizamos la importación de librerías añadiendo las necesarias en el apartado `build-depends` de `mytetris.cabal`.

A screenshot of a text editor showing the 'build-depends' section of a Cabal file. The text is as follows:

```
-- Other library packages from which modules are imported.
build-depends:
    base,
    cairo>=0.9.13,
    gtk >= 0.14.9,
    text,
    random >= 1.2.1.2
```

Figura 2.1. Librerías incluidas en `mytetris.cabal`

En este proyecto se han utilizado diversas librerías con distintos objetivos.

El primer pico a escalar ha sido el del desarrollo de una interfaz gráfica, un concepto no desarrollado en la asignatura y que ha sido de libre aprendizaje. Tras comparar diversos proyectos que desarrollaban entornos gráficos en 2D se eligió: Gtk y Cairo, librerías que juntas facilitan el trabajo de una interfaz agradable visualmente.

Por otro lado, encontramos diversas herramientas: para la programación concurrente y almacenaje de variable en estos entorno, con Control Concurrente y Data.IORef; para la manipulación de listas mediante particiones, con Data.List; y para la creación de números pseudoaleatorios, con System.Random. Todas estas herramientas han sido útiles a la hora de realizar el juego recursivo, desde mostrar piezas aleatorias hasta el control del tablero de juego de una forma más práctica.

3. Código

Se procede a explicar el conjunto del código de una forma superficial y estructurada por diversas ideas que se encuentran en él. Para ello es preciso centrarse en dos contextos independientes que juntos forman el proyecto:

3.1. Bloque físico

La parte esencial de un videojuego por general es su apartado gráfico, pues es la conexión más directa que se tiene con el jugador. En este caso se ha decidido por una interfaz sencilla formada por ventanas.



Figura 3.1. Menú del juego

La primera de ella nos llevará a la gracia del proyecto, mediante un botón. Otro botón a niveles permitirá cambiar la dificultad del juego a base de la velocidad. Además, muestra diversos títulos entre los que se encuentran los créditos. Es una ventana simple que cumple la función de cambio de contexto entre el inicio de la aplicación y el juego.

En la ventana de niveles encontramos una breve descripción de los diversos niveles, que corresponden a números entre 1-9 y siendo un mayor número, una mayor velocidad. Al pulsar el botón aceptar se llevará directamente al juego.



Figura 3.2. Ventana de elección de niveles.

Respecto a la ventana del Tetris, se observa una malla centrada de 10x20, clásica del juego, donde se observan las piezas caer. La función de esta ventana se basa en representar una tabla, o lista de listas, que se encuentra funcional por debajo, en la parte lógica del programa.

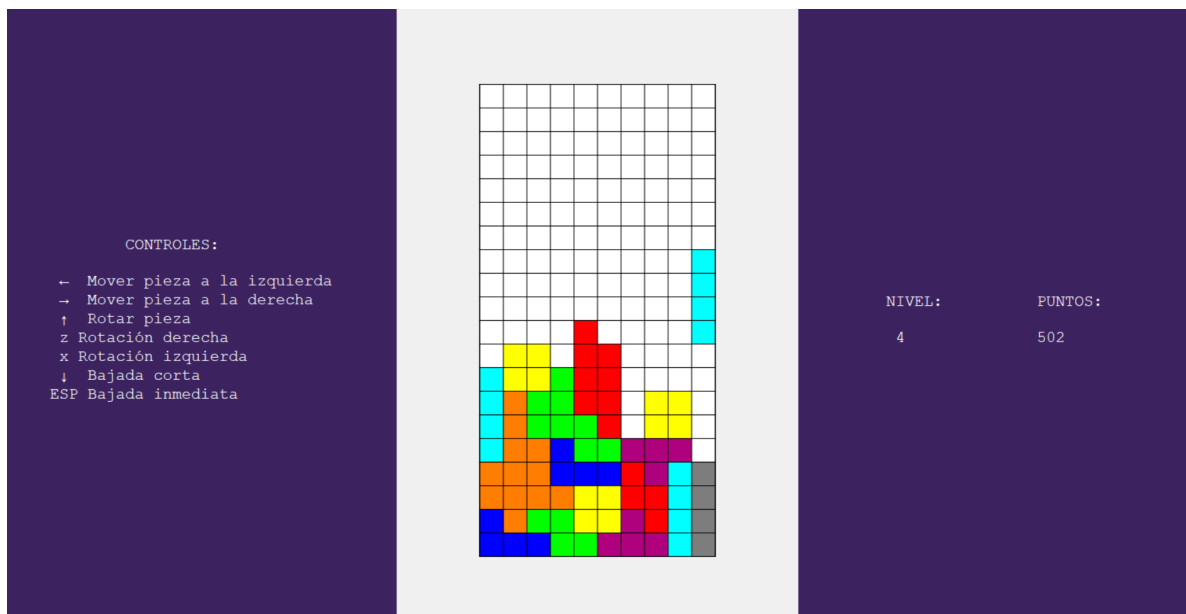


Figura 3.3. Ventana del propio juego

Y por último una ventana de fin de juego. Que surge si un bloque llega al final de la tabla.



Figura 3.4. Ventana fin de juego

3.2. Bloque lógico

Los bloques son parte esencial del tetris, son un total de siete figuras, compuestas por cuatro cuadrados cada una, con distintas formas que se irán acumulando al final de la tabla, pudiendo eliminar parte de ellos si se forma una línea completa. Al ser la parte esencial del juego, también lo es del programa. Por ello se ha centrado la funcionalidad del juego alrededor de ellos.

Siendo el contexto de los bloques el propio tablero, componemos el mismo con celdas que pueden estar vacías o rellenas para formar así los bloques dibujando directamente en ella. A la hora de dibujar se siguen las reglas que deben seguir los bloques, no salirse del tablero y no solaparse. Tarea que se complica con los movimientos que puede realizar los bloques.

Todo se basa en el estado del juego, dato que contiene todos los parámetros necesarios para la realización del juego, por ejemplo tablero, pieza actual, puntos, nivel, etc. Todas las funciones trabajan a partir de este tipo de dato. A continuación vamos a presentar la lógica detrás del juego del Tetris y nuestra adaptación al Haskell.

1. Tablero formados por celdas con celdas del tipo Maybe Int, pudiendo ser Nothing. Gracias a esto, podemos ver todas las piezas, pues donde están colocados estos serán algo y no Nothing.
2. Piezas formadas con una lista de desplazamientos, que dados una posición, identifica la pieza actual.
3. Los movimientos primero miran si en las celdas de la dirección que van a desplazar está ocupada, se borra su rastro anterior y se pintan las nuevas celdas.
4. Cuando se encuentra líneas completas, las elimina y crea nuevas líneas arriba.
5. La velocidad de caída va en sincronía con el nivel que al mismo tiempo depende del puntaje, que aumenta en cada acción de bajada y completación de líneas.

6. Cuando la pieza choca con algo en la altura 0, es decir arriba del todo, finaliza la partida.
7. El orden de las piezas sigue la aleatoriedad original del tetris, es decir, del conjunto de siete piezas aparecerá primero con un orden concreto y en la siguiente iteración el orden cambiará.

Y con toda esta lógica se ha desarrollado el juego.

4. Aprendizaje

Este camino ha sido de puro aprendizaje, desde la instalación de las librerías hasta el resultado final. Se comenzó con la ilusión que se compartió entre compañeros de clase originado en el conocido juego de Tetris. Un juego al que nos apasiona jugar y nos tiene enganchados día sí y día también.

Con la chispa que nos proporcionó esta motivación decidimos comenzar este complejo proyecto. En primer lugar se procedió a investigar sobre cómo comenzar un proyecto de estas dimensiones en Haskell. Para ello se decidió cambiar de entorno de desarrollo a VSCode, pudiendo así tener una mejor comunicación entre los miembros del equipo con la extensión LiveShare.

Continuamos con la implementación de las librerías, una tarea que demoró su tiempo pues al querer instalarlas se requería realizar distintos pasos como la modificación de variables de entorno o la instalación de MSys2, entre otros. Pasos que no estaban claros en un principio, pero que al probar de todo acabó funcionando.

También, nos encontramos con la tesitura de con qué librerías trabajar, pues a parte de buscar aquellas que impulsaran y facilitaran nuestro proyecto, debíamos de encontrar aquellas que no tuvieran conflictos entre sí y no estuvieran obsoletas. Como solución a ello se optó por la investigación de distintos proyectos de juegos recientes que se asemejan al Tetris, para así decidimos por aquellos más utilizados y con mejores resultados, llegando finalmente a las seleccionadas.

Se comenzó a picar código estableciendo una idea base de como se quería estructurar el proyecto. Primero se estableció esa malla que formaría el tablero de los bloques para así después comenzar con el objetivo de hacer caer un bloque. Tarea que se nos atragantó por días y nos llevó a querer cambiar de proyecto, tras la negativa volvió la motivación. Para salir de ese pantano surgió la idea de comenzar a realizar el juego desde la terminal, imprimiendo líneas recursivamente para cada movimiento que se realizará. Sin embargo, uno de los miembros del equipo se quedó con las ganas de hacerlo en una interfaz más visual, y siguió dándole vueltas mientras los otros dos miembros procedían a realizar las bases del juego desde terminal.

Surgió la luz, finalmente el bloque rojo pudo avanzar y el tetris funcionaba en terminal. Así que se procedió a juntar ambas cosas y se terminaron funcionalidades clásicas del juego. Se pudo llegar a una meta digna desde un camino que miramos orgullosos.

5. Conclusiones

Ha sido un proyecto que nos ha permitido conocer esa nueva faceta de Haskell que era más difícil de ver en el resto de prácticas, la conexión con la realidad. Es decir, antes éramos capaces de crear pequeñas funciones aisladas entre sí, sin embargo, ahora hemos conseguido hacer un conjunto de ellas para llegar a un propósito más cercano a lo esperado para cualquier programador, como puede ser un juego. Algo más grande.

6. Bibliografía

Core library de gtk: <https://hackage.haskell.org/package/gtk-0.15.0>

Tutorial Cabal: <https://cabal.readthedocs.io/en/stable/getting-started.html>

Juego de inspiración: <https://tetr.io/>

Puntaje Tetris: <https://tetris.wiki/Scoring>

Colores piezas de Tetris: <https://es.wikipedia.org/wiki/Tetris>