Guion de prácticas

MPALABRADOS (tiles-2)

Mayo 2020

UNIVERSIDAD DE GRANADA

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

Metodología de la Programación

DGIM

Curso 2019/2020

Índice

1. Descripción

En esta práctica se va a desarrollar la última capa de la arquitectura, según el plan de trabajo fiijado en el guión de la Práctica 1. En este caso, se va a implementar la segunda parte de la clase **Tiles**, según la documentación sobre la misma contenida en el fichero **tiles.h** y se va a implementar el cuerpo principal del juego MPALABRADOS. Esta nueva capa de la arquitectura nos va a permitir tener en cuenta cruces válidos o salidas fuera de los márgenes del tablero en su versión definitiva.

2. Práctica a entregar

Se deberá duplicar el proyecto de Netbeans de la práctica anterior y realizar los siguientes cambios (en todos ellos aparece la marca **@warning** avisando de las tareas de implementación que están pendientes).

- Empezar una partida nueva. Para ello los parámetros de llamada serán

  ```
  -l <lang> -w <int> -h <int> [-r <int> -save <matchfile>]
  ```

  especificando el diccionario, el ancho y alto del tablero de juego y, opcionalmente, el número aleatorio y la posibilidad de salvar la la jugada en un fichero con extensión **.match** con la opción `-save`. En caso de que no se indique esta última opción, entonces deberá mostrar en pantalla el estado final de la partida con el mismo formato.

- Continuar una partida existente. Para ello los parámetros de llamada serán

  ```
  -open <matchfile> [-save <matchfile>]
  ```

  indicando la apertura de un fichero **.match** desde el que se restaura el estado anterior de la partida. Opcionalmente, se podrá grabar la partida final si se indica el parámetro `-save` comentado antes.

- **move.h**
  Hasta ahora no se ha diferenciado a nivel interno entre movimientos correctos e incorrectos, pero en esta práctica sí se va a hacer, a dos niveles distintos. A un primer nivel (obligatorio) se deben distinguir los movimientos que son incorrectos poniéndoles un score de -1 con el método **setScore()**. Con esto, la interfaz reconocerá que el movimiento es incorrecto y actuará en consecuencia pues siempre comprueba que $score \leq 0$ en los casos de movimientos erróneos. No obstante, si se quiere dar más información al jugador (opcional) cada vez que se detecte un error, se puede marcar el movimiento con un valor < 0 distinto según el error que se haya producido.

```
#define UNKNOWN             -1   // Error genérico
#define BOARD_OVERFLOW      -2   // La tirada se sale del tablero
#define NONEXISTENT_WORD    -3   // La palabra de la tirada no aparece en el diccionario
#define INFEASIBLE_WORD     -4   // La palabra de la tirada no es compatible com player
#define NOT_FREE            -5   // La palabra empieza en una celda que ya está ocupada
#define MISSING_CROSSWORDS  -6   // La palabra no se cruza con ninguna otra
```

- **tiles.h**
  Añadir el método **findCrosswords()** y, opcionalmente, aunque se sugiere que se implemente, el método **findMaxWord()**.

- **tiles.cpp**
  Leer las reglas 1 al 10 en la sección siguiente, las cuales definen cómo construir las palabras cruzadas a partir de un movimiento. Básicamante, dado un move leído desde el teclado, se debe calcular un movelist en el cada move que contiene uno de los cruces que se producen. En caso de que algún move no sea válido, debido a alguna regla entre las 6 y la 9, su score será negativo reflejando alguna de las siguientes situaciones recogidas en **move.h**

- **window.cpp**
  Es un módulo que contiene dos clases para visualización coloreada del juego, lo más parecido posible al juego real. La clase que nos importa es **Game**, la cual, por simplicidad, absorbe como datos miembro públicos, los datos principales del juego: bag, player, tiles y score, además de los movelist dedicados a registar las buenas tiradas y las incorrectas.

```
class Game : public Window{
public:
  Language language;     // The dictionary, as usual
  Tiles tiles;       // The matrix of letters
  Player player;     // The 7 letters owned by the player
  Bag bag;       // The bag with the remaining letters
  Movelist acceptedmovements, // The list of correct movements
  rejectedmovements,     // The list of bad movements
  crosswords;      // The list used to store the crosswords found
        // after each player turn
  int score,       // Accumulated score
  random;      // Seed of the random generator
```

- Ahora las variables locales de main más importantes ya no se declaran individualmente como en la práctica anterior,

```
int main(int nargs, char * args[]) {
   Bag bag;
   Player player;
   Language language;
   Move move;
   Movelist movements,          /// Original list of movements
            legalmovements,     /// Movements with legal words upon the dictionary
            acceptedmovements, /// Movements accepted in the game
            rejectedmovements; /// Movements not accepted in the game
   Tiles tiles;
   int Id=-1, score=0;
```

  sino que están agrupadas dentro de la clase **Game**

```
int main(int nargs, char * args[]) {
    Move move;
    Game game;
```

  De esta forma, durante todo el juego, cuando se necesite acceder a estos objetos, se deberá a través de game.

```
game.language.setLanguage(lang);
if (game.random>=0)
    game.bag.setRandom(game.random);
game.bag.define(game.language);
game.player.add(game.bag.extract(7-game.player.size()));
game.tiles.setSize(h,w);
game.score = 0;
```

- Introducir la sobrecarga de los operadores << y >> para **Game**. La estructura de las partidas salvadas es la misma que en la práctica 5.

```
MPALABRADOS-V1
30
ES
7 7
. . . . . . .
. . P . . . .
. . O . M . .
. . D . A . .
. . R E M O S
. . A . A S I
. . . . . . .

7 IOPRUUX
74 ONARODRDJD...
```

- **main.cpp**

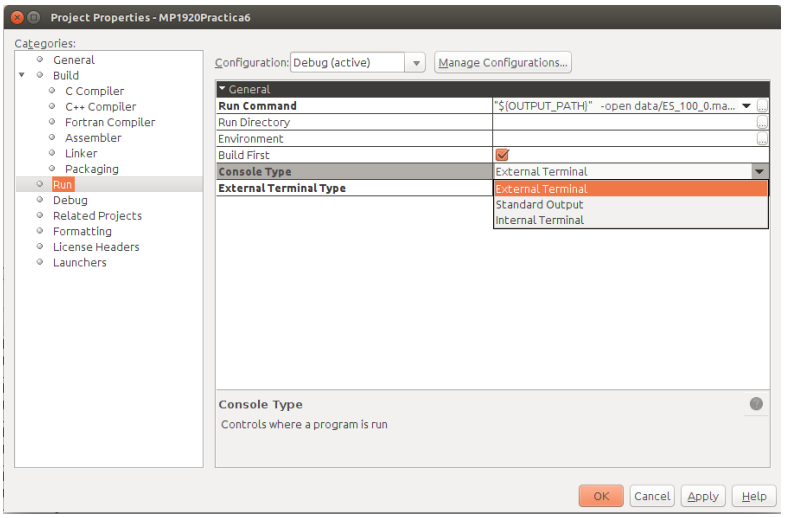
  1. Los parámetros de main son los mismos que en la práctica anterior.
  2. El cuerpo principal del juego ya está programado y se basa EXCLUSIVAMENTE en el uso de los métodos de todas las clases implementadas hasta ahora, en las que recae la principal implementación del juego.

2.1. Configuración de la práctica

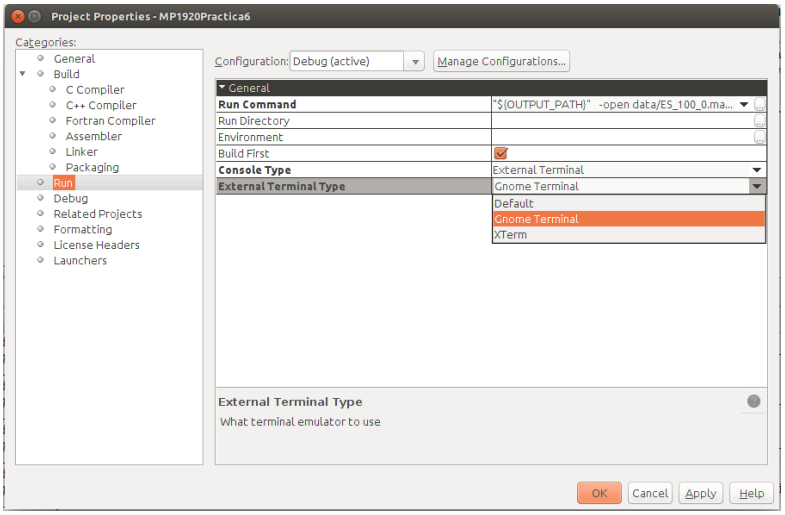
En las versión básica de esta práctica, la configuración es la misma que en la práctica 5, pero para las versiones intermedia y completa, es necesario realizar los siguientes cambios para permitir un mejor control del la interfaz que se construye con los módulos `windows.cpp` y `libansiterminal.a`.

1. Descargar de Prado la librería `Ansiterminal`. Compilar e instalar sus ficheros `AnsiTerminal.h` y `libansiterminal.a` en su carpeta correspondiente.

2. Configurar el proyecto para que se ejecute en una terminal externa, pues la terminal que incorpora Netbeans no soporta los códigos de color de la interfaz. Para ello en Project - Properties - Run es necesario hacer dos cambios. El primero es que el proyecto se ejecute en una terminal externa.



Y el segundo es que esa terminal externa debe ser una consola del sistema, en el caso de Ubuntu, una consola Gnome.



3. Por último, para permitir cambiar, desde dentro del programa, las dimensiones de la ventana de la terminal, es necesatio tener instalado el paquete xterm:

```
sudo apt install xterm
```

2.2. Validación de la práctica

Se debe ejecutar la script **doTests.sh** y comprobar que los resultados que aparecen por pantalla están en verde.

2.3. Entrega de la práctica

Se deberá ejecutar la script **doZipProject.sh** y subir a Prado, en las fechas que se indican en la temporización de la asignatura, el zip resultante, que está almacenado en la carpeta **.zip**/ del proyecto de Netbeans y siempre se llama **MPPractica.zip**.

Modalidad	Puntos	Reglas de cruce Tiles::findCrosswords() Tiles::findMaxWord()	Class Game {...}	
			<< >> doPaint(), setWindowsSize() setCursorOn()/Off()	doConfirmCrosswords() doBadCrosswords() Distinguir errores
Básica	2	X		
Intermedia	0.5	X	X	
Completa	0.5	X	X	X

Cuadro 1: Diferentes modalidades de entrega de las prácticas, sus puntuaciones finales y las diferencias en funcionalidad

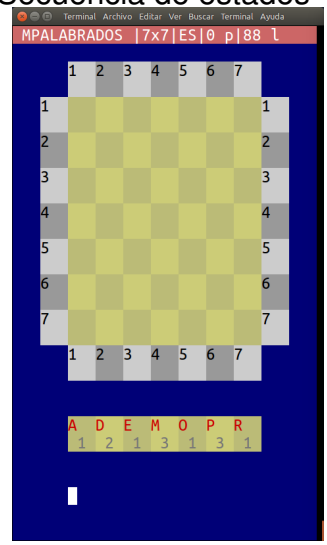
2.4. Modalidades de entrega de la práctica

2.5. Las reglas para jugar a partir de una partida nueva

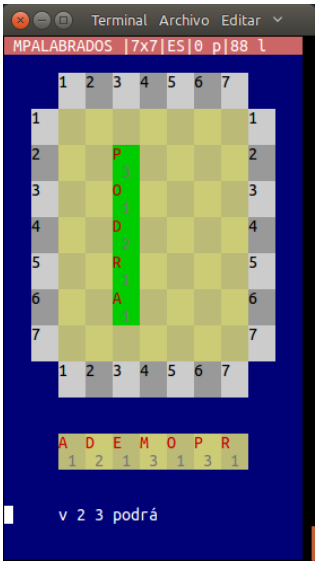
Llamada al programa

```
mp1920practica6 -l ES -r 2020 -w 7 -h 7
```

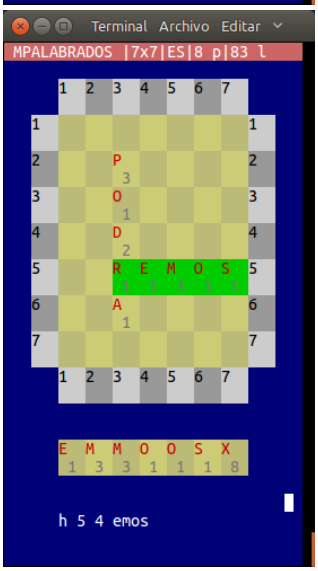
Secuencia de estados



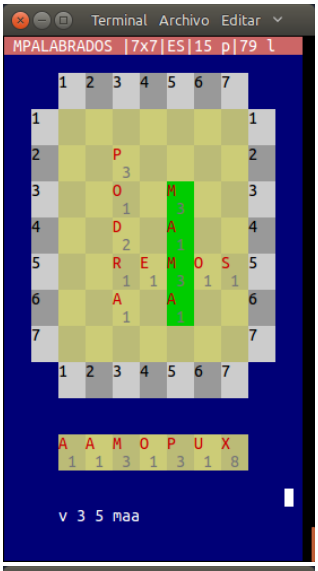
Tablero Inicial



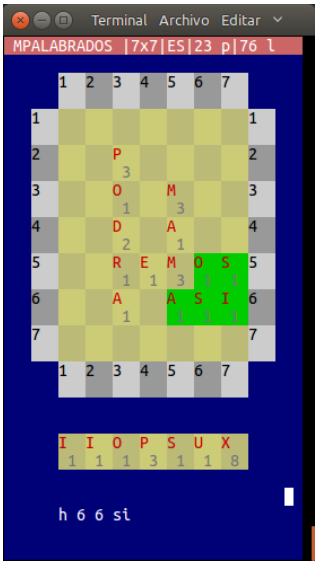
<u>REGLA 1:</u> primera tirada
La primera tirada (**v 2 3 podrá**, normalizado **V 2 3 PODRA**) debe permitir colocar toda la palabra en cualquier parte del tablero, con la única excepción de que deben de caber todas las letras empleadas, sin salir por fuera de los márgenes del tablero. Después de esta tirada, no se podrá poner una palabra que no cruce con ninguna, es decir, a partir de la segunda tirada, una tirada debe cruzarse con, al menos una palabra existente en el tablero. La tirada actual se muestra siempre sombreaada en verde.
<u>move:</u>
mostrado en la figura
<u>crosswords:</u>
V 2 3 PODRA (8)



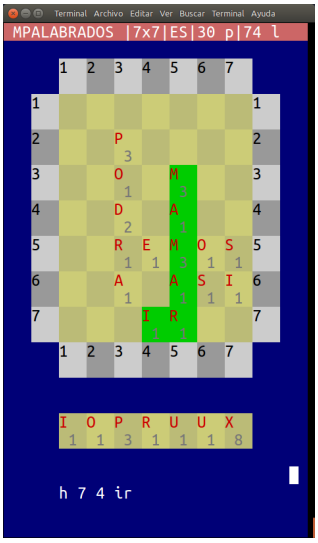
<u>REGLA 2:</u> Cruce al inicio de una palabra
Tirada de las letras **h 5 4 EMOS** en la que se cruza una R al comienzo de la palabra y en la misma dirección, por lo que ambas se combinan formando **REMOS**.
<u>move:</u>
mostrado en la figura
<u>crosswords:</u>
H 5 3 REMOS (7)



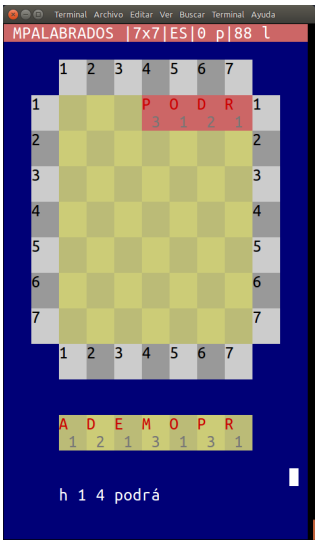
REGLA 3: cruce intermedio
Puede ocurrir que el cruce con letras existentes no suceda al comienzo de una palabara, sino en mitad de la misma. En este caso, la tirada es **MAA** la cual se combina con la **M** existente para formar **MAMA**
move:
mostrado en la figura
crosswords:
V 3 5 MAMA (8)



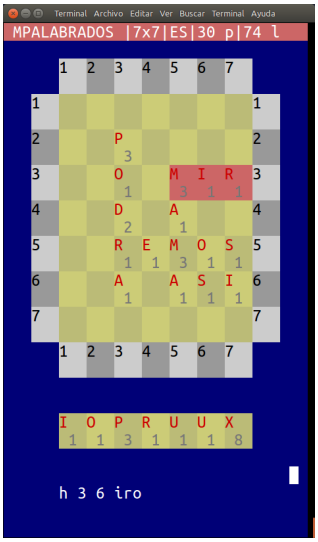
REGLA 4: cruce múltiple
Tirada de las letras **SI** las cuales se combinan con otras tres letras existentes para formar un total de 3 cruces: **ASI**, **OS** y **SI**. Cuando se produce una tirada se deben explorar todos los cruces posibles. Una tirada es posible si todos los cruces son correctos.
move:
mostrado en la figura
crosswords:
V 5 6 OS (2)
V 5 7 SI (2)
H 6 5 ASI (3)



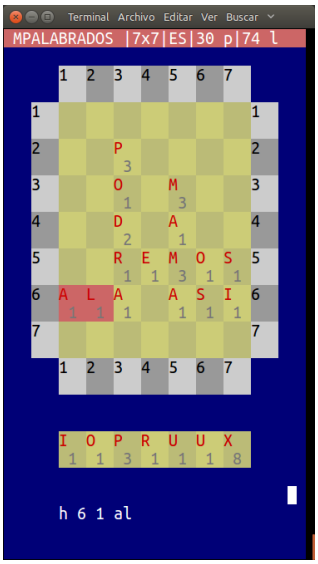
REGLA 5: cruce al final
Otra forma de conseguir un cruce de palabras es colocarlas al final de una palabra completa ya existente. En este caso, al colocar las letras **IR**, además de la tirada **IR**, obtiene la palabra cruzada **MAMAR**
move:
mostrado en la figura
crosswords:
V 3 5 MAMAR (9)
H 7 4 IR (2)



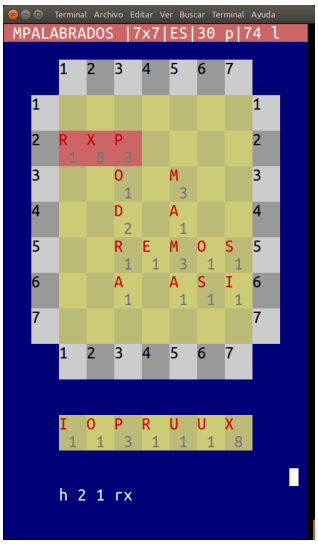
REGLA 6: Desbordamiento del tablero
Aunque la palabra es correcta y está en el diccionario, **H 1 4 PODRA**, la posición elegida hace que no quepa en el tablero, por lo que la tirada no es válida (aparece marcada en rojo)
move:
mostrado en la figura
crosswords:
H 1 4 PODR (BOARD OVERFLOW)



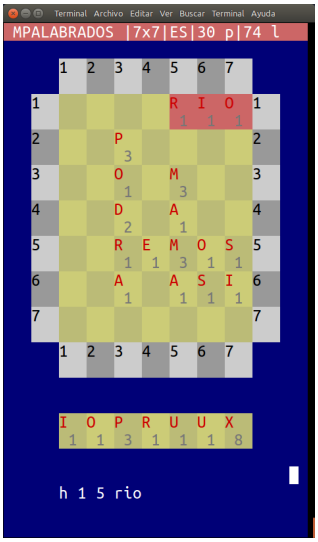
REGLA 6:Desbordamiento del tablero
Aunque la palabra es correcta, está en el diccionario y produce un cruce correcto, **H 3 5 MIRO**, la posición elegida hace que no quepa en el tablero, por lo que la tirada no es válida (aparece marcada en rojo)
move:
mostrado en la figura
crosswords:
H 3 5 MIR (BOARD OVERFLOW)



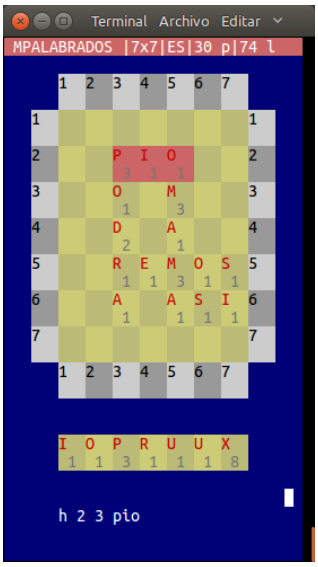
REGLA 7: palabra inviable
La tirada es una palabra correcta del diccionario, pero contiene letras que no están en player. **move:**
mostrado en la figura
crosswords:
H 6 1 AL (INFEASIBLE WORD)



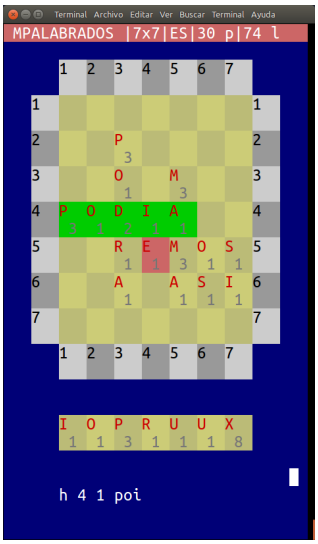
REGLA 8: palabra inexistente
La tirada es válida en tanto en cuanto las letras pertenecían a player, pero la combinación elegida no forma una palabra válida del lenguaje. **move:**
mostrado en la figura
crosswords:
H 2 1 RXP (NONEXISTENT WORD)



REGLA 9: Cruces inexistentes
La tirada no genera ningún cruce con las palabras existentes en el tablero. **move:**
mostrado en la figura
crosswords:
H 1 5 RIO (MISSING CROSSWORDS)



REGLA 10: Celda ocupada
La primera celda de la tirada ya está ocupada, por lo que no se puede añadir al tablero. **move:**
mostrado en la figura
crosswords:
H 2 3 PIO (NOT FREE)



REGLA 11 : Algún cruce mal
Al menos uno de los cruces generados no es un cruce válido según cualquiera de las reglas 6 al 9.
move:
mostrado en la figura
crosswords:
V 4 4 IE (NONEXISTENT WORD)
H 4 1 PODIA (8)