Autores: Iñaki Cadalso, Diego Chicote, Miguel Gomez-Tavira

Documentación técnica *ARL*

En el siguiente documento pasamos a detallar el funcionamiento interno del proyecto de programación que realizamos.

Nuestro juego, ARL(Almost a Rogue-Like), es un juego de exploración de mazmorras y como su nombre indica es “Casi un Rogue-Like”.

Las características más notables de este tipo de juegos, son la muerte permanente del personaje, es decir no se permite salvar la partida, la exploración de mazmorras, el uso de objetos para aumentar las características básicas del personaje, el aumento de experiencia y niveles y por último el combate contra criaturas fantásticas.

Así mismo el argumento de estos juegos no suele ser muy elaborado, y el objetivo final del juego consiste en rescatar a algún personaje o conseguir algún objeto.

Después de está pequeña introducción al tipo de formato del juego pasamos a detallar una descripción de los distintos módulos que componen el programa, así como del funcionamiento del programa en sí mediante estos mismos módulos.

Podemos agrupar los módulos que componen el programa en 4:

- *Elementos:*
  Se encarga de gestionar los diversos personajes, objetos y entornos que intervienen en el juego.

- *Reglas:*
  Se encarga de traducir los datos que introduce el usuario al lenguaje que el programa entiende.

- *Interfaz:*
  Se encarga de mostrar por pantalla los elementos del modulo elementos, así como de actuar de intermediario entre el jugador y el juego.

- *Lector:*
  Se encarga de leer el fichero xml que contiene la información del juego.

El módulo elementos, se compone de los submódulos:

- *objeto.h* / *objeto.c:*
  El módulo objeto contiene la información sobre las estructuras que definen los objetos del juego, así como las funciones para crear, modificar, o eliminar estos.

- *personaje.h* / *personaje.c:*
  El módulo objeto contiene la información sobre las estructuras que definen los personajes del juego, así como las funciones para crear, modificar, o eliminar estos.

Autores: Iñaki Cadalso, Diego Chicote, Miguel Gomez-Tavira

- *inventario.h* / *inventario.c:*
  El módulo inventario contiene la información sobre las estructuras que definen tanto el inventario de los personajes, como de los entornos. Siendo estos listas doblemente enlazadas.

- *pjsEntorno.h* / *pjsEntorno.c:*
  El módulo pjsEntorno contiene la información sobre las estructuras que definen el inventario de personajes de los entornos. Siendo estos listas doblemente enlazadas.

- *entorno.h* / *entorno.c:*
  El módulo entorno contiene la información sobre las estructuras que definen los entornos. Siendo estos listas doblemente enlazadas.

- *mundo.h* / *mundo.c:*
  El módulo mundo contiene la información sobre las estructuras que definen el mundo, que es una lista doblemente enlazada de entorno.

El módulo reglas se compone de los submódulos:

- *acciones.h* / *acciones.c:*
  El módulo acciones se encarga de las funciones que el jugador llama, para realizar acciones en el juego como coger y usar objetos.

- *selección.h* / *selección.c:*
  El módulo selección se encarga de traducir los datos que el jugador pasa al juego, así como de generar la respuesta del juego a esa información recibida.

El módulo interfaz de compone de los submódulos:

- *interface.h* / *interface.c:*
  El módulo interface contiene la información sobre las estructuras que componen la interfaz gráfica.

- *consola.h* / *consola.c:*
  El módulo consola se encarga de las funciones que recogen los datos que el usuario introduce al juego, así como de mostrar por pantalla la respuesta del juego.

- *mapa.h* / *mapa.c:*
  El módulo mapa se encarga de las funciones que se ocupan de mostrar datos a través del panel de mapa de la interfaz, ya sea el mismo mapa, o alguna de las presentaciones.

- *inventarioInterfaz.h* / *inventarioInterfaz.c:*
  El módulo inventarioInterfaz se encarga de las funciones que se ocupan de mostrar datos a través del panel de inventario de la interfaz así como del panel de stats.

Autores: Iñaki Cadalso, Diego Chicote, Miguel Gomez-Tavira

El módulo lector se compone del módulo:

- *Lectorxml.h* / *Lectorxml.c:*
  El módulo Lectorxml se encarga de leer el documento xml que contiene la información del juego y de generar el mundo.

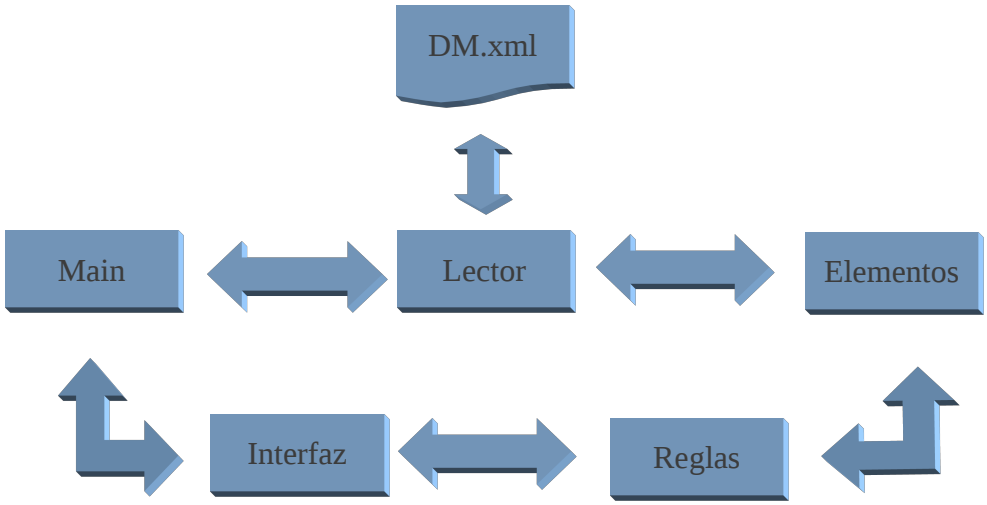
Hay dos módulos adicionales más:

- *tipos.h:*
  Contiene la información sobre los distintos tipos de datos que se usan para el control de errores.

- *main.c:*
  Es el modulo desde el que se carga el juego.

  El funcionamiento global a partir del main es el siguiente:
  1. El main llama al lector xml.
  2. Desde el lector se lee el contenido del fichero y se genera el mundo a través del módulo elementos.
  3. Una vez generado el mundo el main activa la interfaz del juego.
  4. La interfaz captura los datos introducidos por el usuario.
  5. El modulo reglas traduce los datos del usuario al programa. Si procede modifica objetos del modulo elementos. Lanza una respuesta al usuario a través de la interfaz.
  6. Una vez el usuario escriba salir, se destruye la interfaz y el mundo.
  7. A través del main se termina el programa.

El siguiente esquema muestra la comunicación existente entre los distintos módulos que componen la aplicación.



Autores: Iñaki Cadalso, Diego Chicote, Miguel Gomez-Tavira

Por ultimo presentamos una breve guía, para explicar como deben formarse los archivos xml para poder usar el motor de juego, generando un juego diferente, que funcionara con las mismas mecánicas.

El fichero xml está compuesto por la etiqueta principal *<mundo>*, que contiene los diversos entornos del juego. Las distintas etiquetas pueden ser rellenadas intuitivamente al ver su nombre, pero explicaremos aquellas que puedan ser algo más complejas de entender:

Los objetos, cuya etiqueta es *<objeto id="">*, constan de las siguientes etiquetas:

- *<filObjeto>:* Es la fila del mapa en la que aparece el objeto por primera vez.
- *<colObjeto>:* Es la columna del mapa en la que aparece el objeto por primera vez.
- *<nombreObj>:* Es el nombre del objeto, debe ir en mayúsculas para ser reconocido cuando el usuario introduzca su nombre.
- *<descObj>:* Es la descripción del objeto.
- *<graphObj>:* Es el carácter con el que se muestra el objeto por pantalla. Puede ser cualquier carácter ASCII entre 0 y 127.

Los siguientes campos también pertenecen a objeto, son booleanos y por tanto se debe escribir 0 para FALSE y 1 para TRUE:

- *<movil>:* Este campo determina si se puede interactuar con el objeto.
- *<movido>:* Este campo determina si el objeto ha sido desplazado de su posición. Actualmente en desuso.
- *<ilumina>:* Este campo determina si el objeto posee la capacidad de iluminar. Actualmente en desuso.
- *<encendido>:* Este campo determina si un objeto con la capacidad de iluminar está encendido. Actualmente en desuso.
- *<rompible>:* Este campo determina si un objeto puede ser roto por el jugador. Actualmente en desuso.
- *<roto>:* Este campo determina si un objeto rompible ha sido roto. Actualmente en desuso.
- *<teleport>:* Este campo determina si un objeto posee la capacidad de teletransportar al jugador. Actualmente en desuso.
- *<destTeleport/>:* Este campo determina el destino de teletransporte de un objeto con la capacidad de teletransportar. En este caso el campo debe rellenarse con la id del objeto destino. Actualmente en desuso.

Los personajes del entorno cuya etiqueta es *<personaje id="">*, constan de las siguientes etiquetas:
NOTA: No es necesario incluir al personaje principal en el xml, ya que el juego se encarga de generarlo.

- *<filPj>:* Es la fila del mapa en la que aparece el personaje por primera vez.
- *<colPj>:* Es la columna del mapa en la que aparece el personaje por primera vez.
- *<nombrePj>:* Es el nombre del personaje, debe ir en mayúsculas para ser reconocido cuando el usuario introduzca su nombre.
- *<descPj>:* Es la descripción del personaje.

- *<graphPj>:* Es el carácter con el que se muestra el personaje por pantalla. Puede ser cualquier carácter ASCII entre 0 y 127.
- *<invPj>:* Son los objetos que el personaje lleva en su inventario. En la entrega del juego los inventarios de personajes están vacíos.
- *<vitalidad>:* Es la vitalidad de la que dispone el personaje en el momento actual.
- *<vitTotal>:* Es la vitalidad máxima del personaje.
- *<fuerza>:* Es la fuerza del personaje. Este atributo se usa para calcular el daño en combate.
- *<defensa>:* Es la defensa del personaje. Este atributo se usa para calcular la reduccion de daño en combate.

Los siguientes campos son de tipo booleano:

- *<interactuar>:* Este campo indica si se puede hablar con el personaje.
- *<atacable>:* Este campo indica si se puede atacar al personaje.

Los mapas de los entornos cuya etiqueta es *<mapaEntorno>*, constan de las siguientes etiquetas:

- *<colMapa>:* Esta etiqueta determina el numero de columnas de las que consta el mapa.
- *<filMapa>:* Esta etiqueta determina el numero de filas de las que consta el mapa.
- *<salCol>:* Esta etiqueta determina la columna en la que aparece el jugador en el mapa al avanzar de entorno.
- *<salFil>:* Esta etiqueta determina la fila en la que aparece el jugador en el mapa al avanzar de entorno.
- *<enCol>:* Esta etiqueta determina la columna en la que aparece el jugador en el mapa al retroceder de entorno.
- *<enFil>:* Esta etiqueta determina la fila en la que aparece el jugador en el mapa al retroceder de entorno.
- *<mapa>:* En esta etiqueta esta contenido el “dibujo” del mapa del entorno.

Por ultimo ponemos un ejemplo de como debe ser construido un mapa, para poder jugar al juego:

1. Se pinta el mapa en función del valor de las etiquetas mostradas anteriormente.
2. Se añaden los caracteres representativos de los personajes y objetos contenidos en el mapa.
3. Para asegurar el buen funcionamiento debe pintarse como se muestra a continuación, asegurándonos de que los espacios en blanco dentro del área del mapa son espacios y no tabulaciones.

```
<mapa>------------------------------------------------------------
        -                                       -         -
        -  8                                    -         -
        -                               /       -         -
        -                    F                  -----------
        -                                                 -
        -                                                 >
        -     ~~~~~~~~~~~~~                     ------------
        -     ~~~~~~~~~~~~~                     -         -
        -     ~~~~~~~~~~~~~                     -         -
        ------------------------------------------------------------</mapa>
```

NOTA: Los caracteres **<** y **>**, deben ser escritos en el mapa como **<** y **>**, respectivamente. A través de ellos el jugador retrocede o avanza de entorno.