I1: El castillo encantado

Introducción

Esta primera iteración del proyecto (I1) comienza realizando una aventura conversacional sencilla que se desarrolla en un **castillo encantado**. En este caso, hay un jugador, el caballero, que debe conseguir una espada y salir del castillo por el jardín. El castillo consistirá inicialmente en tres espacios: la entrada, el pasillo y el jardín, acorde al mapa de la Figura 1.

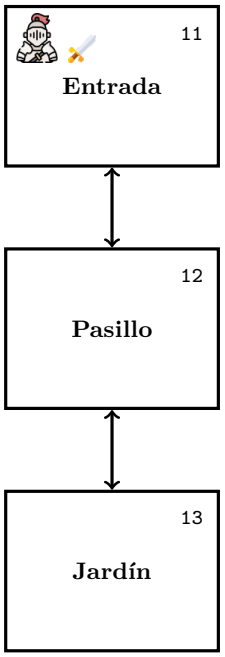

Figura 1: Mapa inicial del castillo.

Con este primer juego se pretenden presentar los conceptos, herramientas y habilidades que se desarrollarán durante el curso.

La Figura 2 ilustra los módulos del proyecto en los que se trabajará en esta iteración inicial. Se trata de la primera aproximación al desarrollo de los módulos esenciales del sistema, como puede verse en el documento de introducción al mismo. En esta ocasión se han representado en **rojo** los módulos que se facilitan implementados de forma rudimentaria. Aunque los módulos representados en rosa no se proporcionan como tales, parte de su funcionalidad está incluida en algunos de los módulos incluidos. Por ejemplo, el módulo GameReader debería incluir todas las funciones que permiten cargar los datos del juego, pero en la versión proporcionada se incluye la función de carga del mapa (espacios) dentro del módulo Game. De forma parecida, el módulo Game maneja un jugador y un objeto primitivos que deberían implementarse en los módulos Player y Object respectivamente, como se hace en el módulo Space con los espacios.

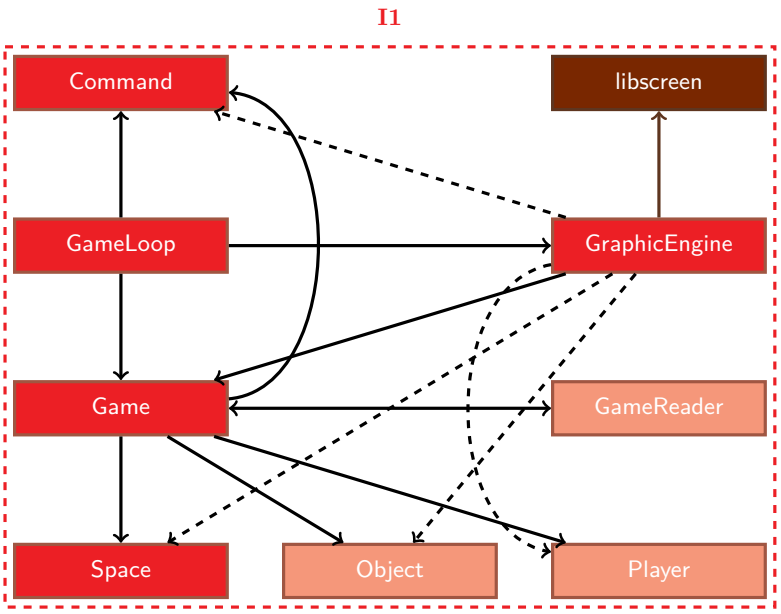

Figura 2: Módulos considerados en la primera iteración (I1) del desarrollo del proyecto.

A pesar de su formato provisional, el material proporcionado se puede compilar y enlazar para generar una aplicación que permite:

1. Cargar mapas de juego (series de espacios enlazados) desde un fichero de datos.
2. Gestionar todos los datos necesarios para la interacción del juego en memoria (espacios y localizaciones para un objeto y un jugador).
3. Soportar la interacción del usuario con el sistema, interpretando comandos para mover al jugador por el mapa y terminar el programa.
4. Mover al jugador por los espacios conectados, haciendo cambiar el estado del juego.
5. Mostrar la posición en cada momento del jugador y de los espacios contiguos al que ocupa, indicando también la ubicación de un objeto, si este se encuentra en algún espacio visible en cada instante.
6. Liberar todos los recursos utilizados antes de terminar la ejecución del programa.

Objetivos

Los objetivos de esta primera iteración del proyecto (I1) son dos. Por un lado, familiarizarse con Linux y el entorno de programación básico de GNU (`gcc` y `make`) y con los fundamentos básicos de estilo de programación y documentación de código. Para ello es necesario revisar los módulos «Sistemas operativos: Linux» y «Compilación de proyectos» del curso SPOC, así como los materiales disponibles en Moodle. Por otro lado, practicar todo lo anterior con el material proporcionado y modificar el mismo para mejorarlo y dotarlo de nuevas funcionalidades.

A continuación se especifican las tareas concretas a realizar en esta iteración.

Requisitos de gestión del proyecto

G1. Adaptar el diagrama de Gantt dado de forma que permita planificar el trabajo de cada equipo. La entrega se realizará al comienzo de la iteración, según lo indicado en Moodle.

G2. Entregar la versión final del cronograma, con los cambios realizados a lo largo de las semanas, para (en caso de necesidad) ajustar los tiempos a la realidad del proyecto.

Requisitos de compilación y entrega

C1. El código debe poderse compilar y enlazar para conseguir un fichero ejecutable.
- Comprobar que no se producen *warnings* al compilar con la opción `-Wall`.

C2. Entregar un `Makefile` que automatice la compilación y enlazado del código semilla proporcionado. La entrega se realizará a mitad de la iteración, según lo indicado en Moodle.

C3. Actualizar el fichero `Makefile` acorde al nuevo código que se implemente. Es fundamental modificarlo a medida que se van añadiendo nuevos ficheros y nuevas dependencias.

Requisitos de documentación y estilo

D1. Documentar **todos** los ficheros proporcionados como código semilla, así como los nuevos que se generen. Nótese que los ficheros `space.c` y `space.h` proporcionados están correctamente documentados y pueden utilizarse como ejemplo para documentar el resto del código. En concreto se debe garantizar:
- Que todas las constantes, variables globales, enumeraciones públicas y estructuras tanto públicas como privadas estén documentadas.
- Que los ficheros fuente incluyan comentarios de cabecera con todos los campos requeridos.
- Que los prototipos de las funciones tanto públicas como privadas estén correctamente documentadas.
- Que las funciones tengan identificado un autor único.
- Que las variables locales de cada módulo o aquellas funciones que precisen explicación estén comentadas.

D2. Mantener un buen estilo de programación en el código entregado, en concreto:
- Que las variables y funciones tengan nombres que ayuden a comprender para qué se usan.
- Que el código esté bien indentado[1].
- Que el estilo sea homogéneo en todo el código[2].

[1] La indentación deberá ser homogénea. Todos los bloques de código pertenecientes a un mismo nivel deberán quedar con la misma indentación. Además, deberán usarse caracteres de tabulación o espacios (siempre el mismo número de espacios por nivel), pero nunca mezclar tabulación y espacios.

[2] Como mínimo debe cumplirse lo siguiente: que los nombres de las funciones comiencen con el nombre del módulo; que las variables, funciones, etc., sigan convención *camel case* o *snake case*, pero nunca mezcladas; que el estilo de codificación sea siempre el mismo (p.e. *K&R*, *Linux coding conventions*, etc.), pero nunca mezclar estilos.

Requisitos de funcionalidad

F1. Corregir los posibles errores que tenga el código dado y que generan *warnings* de compilación. Una forma de compilar rápidamente el código para poder probarlo es la siguiente[3]:

```
$ gcc -o castle *.c -lscreen -L.
```

F2. Crear el módulo GameReader (ficheros `game_reader.c` y `game_reader.h`) extrayendo la funcionalidad ya existente en el módulo Game necesaria para la carga de los espacios. El nuevo módulo creado deberá incorporar, en futuras iteraciones, los lectores de otros elementos necesarios para los juegos (objetos, jugadores, etc.).

F3. Modificar los módulos pertinentes para sustituir la funcionalidad de lectura de espacios por la misma incluida en el nuevo módulo GameReader.

F4. Crear el módulo Object (ficheros `object.c` y `object.h`) que integre la funcionalidad necesaria para el manejo de objetos, de forma similar a como se hace en el módulo Space para gestionar la de los espacios. En particular, los objetos deberán implementarse como una estructura de datos con un campo de identificación (`id`) y otro con el nombre del objeto (`name`), así como facilitar las funciones necesarias para crear y destruir objetos (`create` y `destroy`), leer y cambiar los valores de sus campos (`get` y `set`) e imprimir (`print`) el contenido de los mismos (por ejemplo, para su depuración).

F5. Crear el módulo Player (ficheros `player.c` y `player.h`) que integre la funcionalidad necesaria para el manejo de jugadores (también de forma parecida a como lo hace el módulo Space para los espacios). En particular, los jugadores deberán implementarse como una estructura de datos con campos de identificación (`id`), nombre (`name`), localización (`location`, para el identificador del espacio donde se encuentran) y objeto portado (`object`, para el identificador del objeto que lleva). Además, se deben incluir las funciones necesarias para crear y destruir jugadores (`create` y `destroy`), leer y cambiar los valores de sus campos (`get` y `set`) e imprimir el contenido de los mismos (`print`), como en el caso anterior.

F6. Modificar los módulos existentes para que utilicen los nuevos objetos y jugadores sustituyendo la funcionalidad correspondiente integrada inicialmente. Por ejemplo, implementar la funcionalidad del módulo Game sustituyendo los identificadores de `player_location` y `object_location` por sendos punteros a las nuevas estructuras de Player y Object, y utilizar las funciones adecuadas de los nuevos módulos para la manipulación de datos necesaria. O, por ejemplo, modificar el módulo Space para que la estructura de datos de los espacios guarde el identificador del objeto que contienen estos en lugar de una variable booleana, y preparar las funciones necesarias para manipular dicho valor.

F7. Crear dos nuevos comandos que permitan al jugador coger el objeto de un espacio (`take` o `t`) y dejar el objeto que lleve el jugador en el espacio en el que se encuentre (`drop` o `d`), considerando que solo hay un objeto en el juego.

F8. Crear un nuevo fichero de carga de espacios, modificando el castillo dado: crear tres caminos, uno con 6 espacios que llegue hasta el jardín, otro con 3 espacios que salga del tercer espacio y un último, que salga del segundo espacio y llegue a una mazmorra sin salida, como se muestra en la Figura 3. Obsérvese que, con los comandos actuales, no se pueden alcanzar los caminos alternativos (y no se podrán alcanzar hasta la siguiente iteración).

[3] Este comando sirve para empezar a trabajar en el proyecto, pero lo adecuado es revisar y entender el contenido del SPOC para adaptarlo al proyecto.

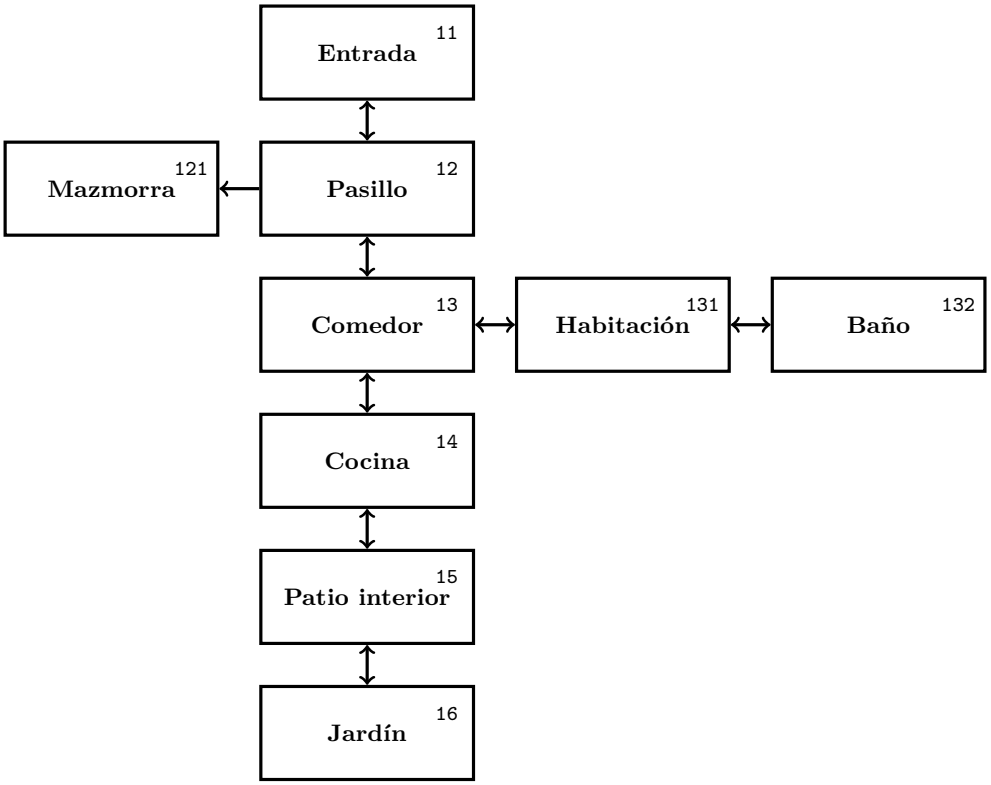

Figura 3: Mapa del castillo para el Requisito F8.

Criterios de corrección

La puntuación final de esta práctica forma parte de la nota final en el porcentaje establecido al principio del curso para la I1. En particular, la calificación de este entregable se calculará siguiendo la rúbrica de la Tabla 1, en la que la segunda columna muestra la puntuación de cada requisito, y las tres últimas columnas incluyen una sugerencia de con qué nivel de completitud hay que abordar cada requisito para alcanzar la calificación indicada.

Tras la entrega de la iteración se realizará una *prueba individual* para demostrar la adquisición de conceptos fundamentales de esta parte del proyecto. Si se supera dicha prueba, la calificación final de la iteración será:

$$\texttt{Final\_I1} = \text{máx}\left\{10{,}0 \;;\; \texttt{Entregable\_I1} + \frac{\texttt{Prueba\_I1} - 5{,}0}{5{,}0}\right\}.$$

Además, no se podrá aprobar la iteración en los siguientes casos extremos:

- Se suspende la prueba individual.
- No se ha completado el diagrama de Gantt inicial.
- El código entregado no compila.
- El código entregado carece de cualquier documentación, o de un estilo de codificación razonable.
- No se ha implementado ningún requisito de funcionalidad, o el código modificado no mantiene la funcionalidad básica del código semilla.

Tabla 1: Rúbrica para el entregable de la primera iteración (I1).

Objetivo	Puntuación	Aprobado	Notable	Sobresaliente
G1	0,5			
G2	0,5			
C1	1,0			
C2	0,5			
C3	1,0			
D1	1,5			
D2	1,0			
F1	0,5			
F2	0,5			
F3	0,5			
F4	0,5			
F5	0,5			
F6	0,5			
F7	0,5			
F8	0,5			

Otras consideraciones

- No se puntuarán aquellos requisitos de funcionalidad que no se puedan probar al ejecutar el juego.