Primera Iteración del Proyecto (I1): Juego de la Oca Básico

Introducción

Como se ha explicado en la introducción al Juego de la Oca, este juego puede verse como una Aventura Conversacional básica y por eso se va a utilizar como referencia para las dos primeras iteraciones del desarrollo del proyecto de la asignatura. Pretendemos demostrar de esta forma el interés de los conceptos, herramientas y habilidades que se desarrollarán durante el curso.

La Figura 1 ilustra los módulos del proyecto en los que se trabajará en esta iteración inicial (I1). Se trata de la primera aproximación al desarrollo de los módulos esenciales del sistema, como puede verse en el documento de introducción al mismo (`I0-IntroProyecto.pdf`).

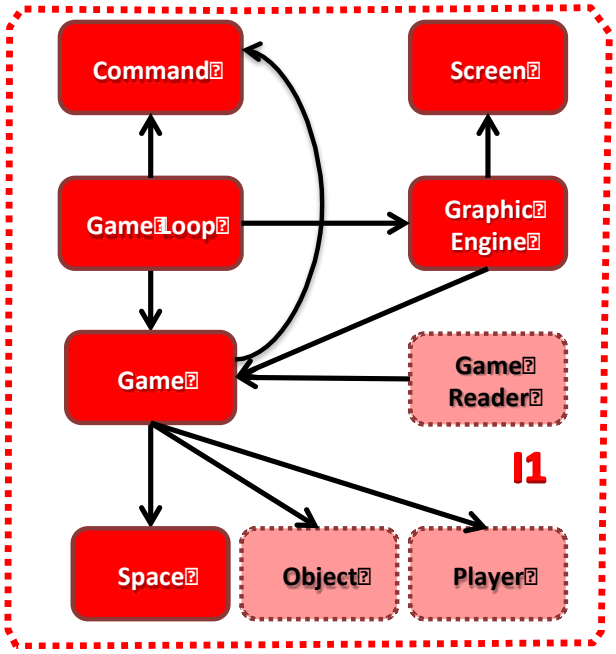

Figura 1. Módulos considerados en la primera iteración (I1) del desarrollo proyecto.

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 rojos. 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 tablero (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.

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

1. Cargar tableros de juego (series de espacios enlazados secuencialmente) desde un fichero de datos.

2. Gestionar todos los datos necesarios para la interacción del juego en la memoria del equipo (espacios y localizaciones en el tablero para un objeto y un jugador).
3. Soportar la interacción del usuario con el sistema, interpretando comandos para mover al jugador por el tablero y terminar el programa.
4. Mover al jugador por los espacios conectados longitudinalmente, haciendo cambiar el estado del juego.
5. Mostrar la posición en cada momento del jugador y de las casillas contiguas a la que ocupa, indicando también la ubicación de un objeto, si este se encuentra en alguna de las casillas visibles 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 el entorno de programación básico de GNU (`gcc` y `make`) y con los fundamentos básicos de estilo de programación y documentación de código, así como iniciarse en el empleo de herramientas esenciales para el control de versiones. Por otro lado, practicar todo lo anterior con el material proporcionado y modificar el mismo para mejorarlo y dotarlo de nuevas funcionalidades.

Las mejoras y modificaciones requeridas que serán objeto de evaluación de esta iteración son las siguientes:

1. Crear los ficheros `Makefile` que automaticen la compilación y enlazado del código proporcionado y del nuevo código que se implemente.
2. Corregir el estilo de programación y documentar tanto los ficheros fuente proporcionados como semilla como los nuevos que se generen.
3. Crear el módulo `GameReader` 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.).
4. Modificar el módulo `Game` proporcionado para sustituir la funcionalidad de lectura de espacios por la misma incluida en el nuevo módulo `GameReader`.
5. Crear el módulo `Object` 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 y otro con el nombre del objeto, 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).
6. Crear el módulo `Player` 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, nombre, localización (para el identificador del espacio donde se encuentran) y objeto portado (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.

7. 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 éstos, y utilizar las funciones necesarias para manipular dicho valor.
8. Crear dos nuevos comandos que permitan al jugador coger un objeto de un espacio y dejar el objeto que lleva en un espacio (normalmente distinto del anterior), considerando que, de momento, el jugador sólo puede llevar un objeto y que en los espacios no puede haber más de un objeto a la vez.
9. Crear un nuevo fichero de carga de espacios para implementar un tablero para el Juego de la Oca con 12 casillas, todas enlazadas secuencialmente (norte/sur) y dos concretas (la 5 y la 9, que serán ocas) que estarán enlazadas de forma especial (este/oeste).
10. Modificar el programa (`game_loop`) principal para que utilice los nuevos tipos de datos y las nuevas funcionalidades.
11. Modificar el `Makefile` inicial para automatizar la compilación y el enlazado del conjunto.
12. Depurar el código hasta conseguir su correcto funcionamiento.

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 según los siguientes criterios:

- **C**: Si se obtiene C en todas las filas de la tabla de rúbrica.
- **B**: Si se obtienen, al menos, dos Bes y el resto Ces. Excepcionalmente sólo con una B.
- **A**: Si se obtienen, al menos, dos Aes y el resto Bes. Excepcionalmente sólo con una A.

Cualquier trabajo que no cumpla los requisitos de la columna C obtendrá una puntuación inferior a 5.

Tabla de rúbrica:

	C **(5-6,9)**	**B** **(7-8,9)**	**A** **(9-10)**
Entrega y compilación	(a) Se han entregado en el momento establecido todos los ficheros solicitados. y (b) Es posible compilar y enlazar los fuentes para conseguir un ejecutable.	Además de lo anterior: Es posible compilar y enlazar los fuentes de forma automatizada utilizando el `Makefile` entregado.	Además de lo anterior: La compilación y el enlazado no producen errores ni avisos (*warnings*) utilizando la opción `-Wall`.
Funcionalidad	El usuario dispone de la funcionalidad básica incluida en la versión proporcionada en el código semilla tras haberse realizado las modificaciones pedidas, pero el resto de funcionalidad no está disponible.	Además de lo anterior: Se dispone de algunas de las nuevas funcionalidades solicitadas, pero no la totalidad de ellas.	Se dispone de todas las nuevas funcionalidades solicitadas.
Estilo y documentación	(a) Que las variables y funciones tengan nombres que ayudan a comprender para qué se usan. y (b) Que todas las constantes, variables globales, enumeraciones públicas y estructuras públicas se hayan documentado. y (c) Que el código esté bien indentado[1].	Además de lo anterior: (a) Que los ficheros fuente incluyan comentarios de cabecera con todos los campos requeridos. y (b) Que las interfaces de los prototipos de las funciones tanto públicas como privadas estén correctamente documentadas. y (c) Que las funciones tengan identificado un autor único.	Además de lo anterior (a) Que el estilo sea homogéneo en todo el código[2]. y (b) Que las variables locales a cada módulo o función que precisen explicación se hayan comentado.

[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 de codificación.