Cuarta Iteración del Proyecto (I4): Juego Conversacional

Introducción

En esta cuarta iteración (I4) se debe completar el desarrollo del proyecto, empleando para ello conceptos, habilidades y herramientas en los que se ha trabajado durante el curso. En esta iteración se completarán las funcionalidades básicas del sistema para soportar Aventuras Conversacionales, y se añadirán otras que se consideren adecuadas para el Juego Original que cada equipo deberá diseñar e implementar sobre el sistema.

La Figura 1 ilustra los módulos del proyecto en los que se trabajará en la I4, partiendo del material elaborado en las anteriores iteraciones (I1, I2 e I3). En esta ocasión se completará el desarrollo de los módulos fundamentales del sistema y se añadirán otros con funcionalidades especiales convenientes para el juego particular que se implemente.

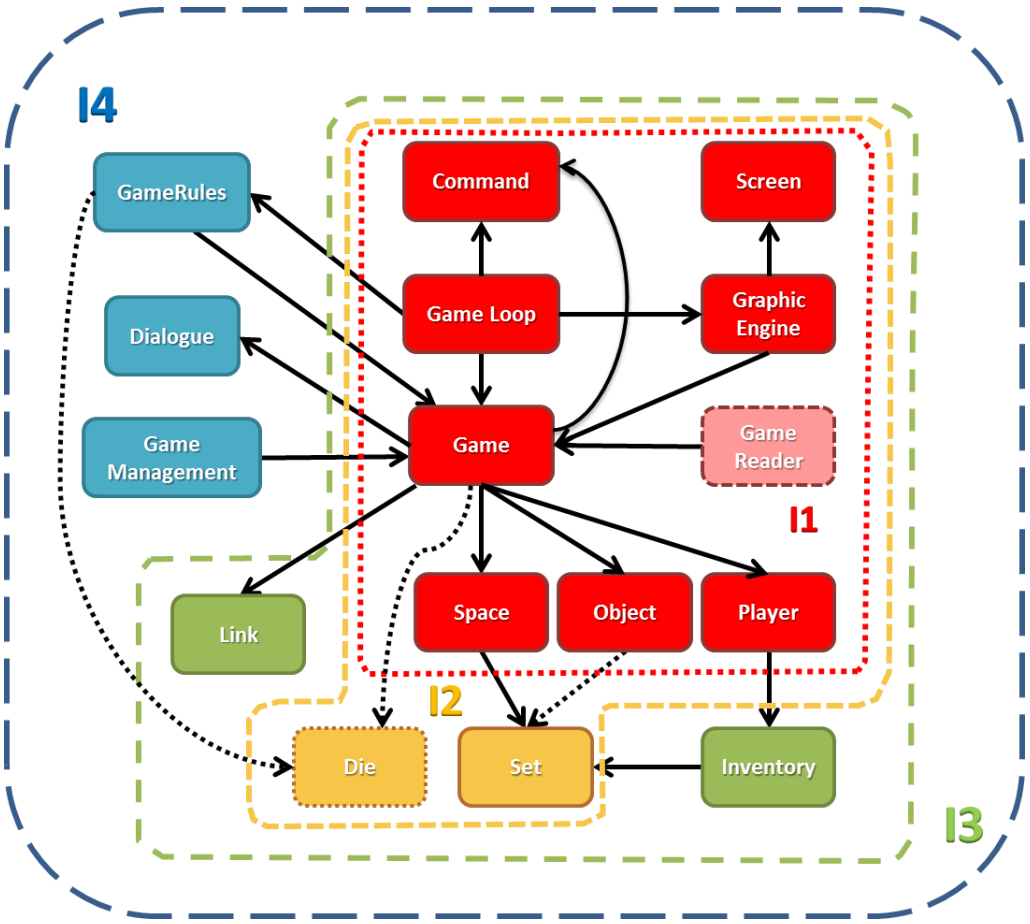

Figura 1. Módulos considerados en la cuarta iteración (I4), correspondiente al final del desarrollo proyecto.

Los módulos obtenidos como resultado de la I1 se han representado en rojo en la Figura 1. En ocre se presentan los módulos desarrollados en la I2. En verde se muestran los módulos que se desarrollaron en la I3. Por último, en azul se representan algunos ejemplos de módulos adicionales que se desarrollarán en la I4, en la que además deberán ampliarse y emplearse algunos de los módulos de anteriores iteraciones (rojos, ocres y verdes).

Los ejemplos de nuevos módulos incluidos en la Figura 1 son tres. El primero, `GameManagement` (Gestor de Juego), tiene la finalidad de dotar a la plataforma de la capacidad de gestionar partidas. El segundo, `Dialogue` (Diálogo), está destinado a dotar al sistema de una interfaz más amigable y natural. Por último, el tercero, `GameRules` (Reglas de Juego), está dirigido a dotar al juego de reglas que afecten a sus cambios de estado en cada momento, además de los cambios que la interacción del jugador provoque. De todos estos módulos se hablará con más detalle a lo largo de este documento.

El material de partida, resultado de I1, I2 e I3, debería generar una aplicación que permita:

1. Cargar espacios, enlaces, objetos y jugadores desde un fichero de datos.
2. Gestionar todo lo necesario para la implementación de juegos conversacionales básicos utilizando todos los elementos mencionados en el punto anterior, como podría ser la Oca.
3. Soportar la interacción del usuario con el sistema, interpretando comandos para mover al jugador, manipular objetos, inspeccionar espacios y objetos, y salir del programa (además de utilizar un dado).
4. Soportar la generación de un fichero con el registro de comandos ejecutados y los correspondientes resultados obtenidos.
5. Mostrar el estado del juego en cada momento: posición del jugador, descripciones del espacio donde está, ubicación de los objetos en el mapa (tablero) del juego, objetos en el inventario del jugador, así como el último valor del dado.
6. Liberar todos los recursos utilizados antes de terminar la ejecución del programa.

Como resultado de la I4, se espera:

1. Una aplicación (`JuegoConv`) que permita implementar una Aventura Conversacional utilizando el sistema desarrollado a lo largo del curso y las extensiones incorporadas en esta última iteración. El programa deberá utilizar todas las características que debería proporcionar el material de partida (resultado de las anteriores iteraciones) convenientemente corregido, mejorado y ampliado.
2. Una aventura que contenga, al menos, 10 espacios y 20 objetos, con su correspondiente fichero de datos (incluyendo espacios, enlaces, objetos, jugador, etc.).
3. Corrección, mejora y ampliación de las funcionalidades de la plataforma de partida de acuerdo con los requerimientos que se indiquen más adelante y las necesidades particulares de la aplicación y la aventura del punto anterior.
4. Documentación de usuario que incluya la información necesaria para utilizar el juego, un mapa, indicaciones para ir del principio al final del juego de forma que se aprecien todas las características y funcionalidades implementadas, y un fichero de comandos para seguir dicho camino.
5. Programas de prueba de cada módulo (`*_test.c`) y documentación de pruebas de todos los módulos (documentos de diseño e informes de pruebas).
6. `Makefile` que automatice la gestión del proyecto: (a) parámetros de compilación exigentes (`-Wall -ansi -pedantic`) y de depuración (`-g`); (b) compilación/enlazado de cada módulo junto con su programa de prueba (`*_test`); (c) gestión de ficheros (por lo menos borrado de objetos y de ejecutables).
7. Documentación técnica del proyecto en formato HTML generada con Doxygen.

8. Gestión del proyecto a lo largo de la I4, con diagramas de Gantt y actas de las reuniones celebradas.

Objetivos

El objetivo de esta cuarta iteración del proyecto (I4) es poner en práctica todo lo aprendido durante el curso sobre trabajo en equipo, gestión de proyectos, uso y diseño de bibliotecas, programación, pruebas, depuración, documentación, etc.

Las mejoras y modificaciones requeridas (requisitos R1, R2, etc.), así como las actividades y las tareas que se deben llevar a cabo son las siguientes:

1. [R1] Modificar el módulo `Space` para que permita que los espacios puedan estar iluminados o no, de modo que sólo puedan verse las descripciones y los objetos de los espacios iluminados y sea necesario iluminarlos de alguna manera para acceder a tal información en los juegos. Para ello debería añadirse un nuevo campo de tipo booleano a la estructura de datos correspondiente e implementarse las funciones necesarias para manipular sus valores (como `set` y `get`) e imprimir el contenido del mismo para su depuración (`print`).
2. [R2] Modificar de nuevo el módulo `Space` para que incorpore, además de las conexiones hacia los cuatro puntos cardinales (`north`, `east`, `south` y `west`) otras dos hacia arriba y abajo (`up` y `down`), para mapas con varios niveles (pisos). Implementar las funciones necesarias para manipular e imprimir dichos campos.
3. [R3] Modificar el módulo `Space` otra vez para que incluya una nueva descripción más detallada que la existente, de modo que la antigua se use para describir el espacio actual cuando se muestra el estado del juego y la nueva cuando se examina el mismo ("`inspect space`"). Implementar las funciones necesarias para manipular e imprimir el nuevo campo.
4. [R4] Modificar el comando de examinar el espacio donde está el jugador implementado en I3 ("`inspect space`") para que muestre la nueva descripción detallada si el espacio está iluminado, o nada en caso contrario.
5. [R5] Modificar el módulo `Object` para que incorpore soporte para nuevas propiedades junto con las funciones necesarias para su manipulación e impresión:
    a. Movible (`movable`), para indicar si el objeto se puede mover de su ubicación. Para ello debería añadirse un nuevo campo de tipo booleano a la estructura de datos correspondiente. Por defecto los objetos NO se pueden mover. El jugador solo podrá coger objetos para su inventario si éstos son movibles.
    b. Movido (`moved`), para señalar si el objeto movible se ha movido de su ubicación original en algún momento (es decir, si se ha cogido). Para ello debería añadirse un nuevo campo de tipo booleano a la estructura de datos correspondiente. Por defecto los objetos no se han movido. El contenido de este campo se ignorará si el objeto no se puede mover.
    c. Oculto (`hidden`), para indicar si el objeto está oculto. Por omisión los objetos son visibles. En el espacio donde se encuentra el jugador sólo se mostrarán los objetos visibles, aunque los ocultos se podrán manipular normalmente (coger, dejar, examinar, etc.). Además, sólo se mostrará la ubicación de los objetos que no estén ocultos.

   d. Abre (`open`), para establecer si el objeto puede abrir un determinado enlace (`link`) especificado por su `Id`. Por omisión no puede abrir nada, en cuyo caso el campo tendrá el valor `NO_ID`.
   e. Ilumina (`illuminate`), para señalar si el objeto puede iluminar un espacio. Para ello debería añadirse un nuevo campo de tipo booleano a la estructura de datos correspondiente, que será `TRUE` cuando el objeto pueda iluminar y `FALSE` en caso contrario. Por defecto los objetos no iluminan.
   f. Encendido (`turnedon`), para establecer si un objeto que puede iluminar está encendido o no. Se trata de un campo booleano, que por defecto tendrá el valor `FALSE` y que sólo se usará cuando el objeto pueda iluminar.
6. [R6] Modificar el módulo `Object` otra vez para que incorpore, además de la descripción ya existente, otra alternativa para cuando el objeto se haya movido. Implementar las funciones necesarias para manipular e imprimir dicho campo con los demás.
7. [R7] Modificar el comando de examinar objetos ("`inspect <obj>`", donde `<obj>` es el nombre de un objeto) para que muestre la descripción adecuada dependiendo de si el objeto se ha movido o no, en caso de que el espacio esté iluminado.
8. [R8] Crear los comandos necesarios para encender y apagar objetos que se pueden iluminar ("`turnon <obj>`" y "`turnoff <obj>`", donde `<obj>` es el nombre de un objeto, por ejemplo "`turnon lantern`" o "`turnoff torch`") de manera parecida a como se hacía en iteraciones anteriores para coger objetos o moverse entre espacios.
9. [R9] Añadir un nuevo comando para abrir enlaces con objetos ("`open <lnk> with <obj>`", donde `<lnk>` es el nombre de un enlace y `<obj>` el de un objeto, por ejemplo "`open door with key`" o "`open wall with tnt`") partiendo del procedimiento seguido en comandos anteriores.
10. [R10] Modificar, en caso de necesidad, la parte del fichero de datos correspondiente a los módulos alterados y todos los módulos afectados por los cambios realizados, por ejemplo, las funciones de carga de espacios y objetos desde ficheros.
11. [R11] Crear una aventura que incluya 10 espacios y 20 objetos, por lo menos, con su argumento y su correspondiente fichero de datos (incluyendo espacios, enlaces, objetos, jugador, etc.).
12. [R12] Crear una documentación de usuario que incluya:
   a. Toda la información necesaria para utilizar el juego.
   b. Un mapa del mismo con espacios, enlaces y ubicación de los objetos, similar al ejemplo incluido para la Oca en el enunciado de la I3.
   c. Las indicaciones para ir desde el inicio hasta un final del juego por un camino donde se pongan de manifiesto todas las características y funcionalidades implementadas.
   d. Un fichero con la lista de comandos para seguir el camino anterior. Un ejemplo de este tipo de archivo podría ser:

```
move north

move west

inspect space

inspect lantern
```

```
take lantern

move west

drop lantern

…

exit
```

13. [R13] Implementación de un módulo `GameManagement` (Gestor de Partida). Ahora, el módulo `GameReader` pasará a llamarse `GameManagement`, el cual no sólo se encargará de cargar los datos de un juego, sino también de guardar el estado actual de una partida para poder recuperarla más adelante. Con este fin, se propone que este módulo conste de una función para salvar (`game_management_save`) y otra para cargar (`game_management_load`). La función `game_management_save` almacenará en un fichero (cuyo nombre deberá parametrizarse) el estado actual de la partida, esto es, el contenido de la estructura `Game`. Con respecto a `game_management_load`, el objetivo no es otro que actualizar la partida, es decir, cargar el contenido de la estructura `Game` de acuerdo con el contenido del fichero cuyo nombre deberá pasarse como argumento a la función de carga.
14. [R14] Añadir dos nuevos comandos para permitir al usuario salvar y cargar partidas. El comando para salvar una partida (`save`) deberá también permitir indicar el nombre del fichero donde se guardarán los datos de la estructura `Game`. El comando para cargar una partida (`load`) deberá permitir indicar el nombre del fichero de configuración con los datos que tiene actualmente la partida a cargar.
15. [R15] Crear un módulo `Dialogue` (Diálogo). A fin de simular una interacción más amigable entre el usuario y el computador durante la ejecución de la aventura conversacional, se pide la implementación de este módulo, de manera que:
    a. Para cada comando ejecutado por el usuario, se acabe mostrando una frase indicativa de qué acción ha realizado y si ésta se ha llevado a cabo con éxito o no. Por ejemplo, si el usuario invoca el comando "`move west`" y la ejecución ha sido correcta, el juego podría construir y mostrar una frase del tipo "`You've moved west. Now you are in <space_description>.`", donde `<space_description>` es la descripción del espacio. Si el comando no ha podido llevarse a cabo satisfactoriamente, podría producir un mensaje del tipo "`You can not move west. Try another action.`".
    b. Asimismo, se deberá identificar si el usuario ha ejecutado de forma consecutiva un mismo comando sin éxito o si ha intentado ejecutar un comando que no existe. Así, podrá responder al primer caso con un mensaje del tipo "`You have done this before without success.`" y en el segundo caso con un mensaje como "`This is not a valid action. Try again.`".

    Se podrán añadir todas las reglas de diálogo que se consideren oportunas, con un mínimo de una regla de diálogo por cada posible comando de usuario, y una o varias reglas para tratar las repeticiones de comando y otras tantas para la detección de comandos nulos.

16. [R16] Implementar un módulo `GameRules` (Reglas de Juego). A fin de dar cierto carácter no determinista a la ejecución del juego, además de las acciones del usuario, se implementarán acciones que sólo podrá ejecutar el juego. Así, basándose en el módulo `Command` con los comandos del usuario, deberá implementarse el módulo `GameRules` donde deberían añadirse las acciones que podrá ejecutar el juego sin intervención del usuario, tales como iluminar o dejar sin luz algunos espacios, cerrar o abrir ciertos enlaces, cambiar los enlaces de algún espacio, ocultar o cambiar de ubicación un objeto, etc. Para ello, tras ejecutar un determinado número de instrucciones del usuario, se ejecutará al azar (empleando para ello el dado `Die` disponible en `Game`) una de las acciones (reglas) del juego. Deberá añadirse una regla especial que sea `NO_RULE` de modo que no siempre se ejecuten reglas de juego. Se podrán añadir tantas reglas de juego como se consideren oportunas, con un mínimo de seis.
17. Además de los requisitos listados anteriormente, se deberán llevar a cabo las siguientes actividades y tareas:
    a. **Modificar, en caso de necesidad, todos los módulos afectados por los cambios realizados**, para mantener la funcionalidad previa e incorporar la nueva pretendida.
    b. **Implementar y/o completar los bancos de pruebas**, así como los informes de pruebas de todos los módulos.
    c. **Modificar el `Makefile`** del proyecto considerando los nuevos módulos implementados, con el fin de mantener la automatización de la compilación y del enlazado del proyecto completo.
    d. **Depurar el código** hasta conseguir su correcto funcionamiento con los módulos modificados y los nuevos implementados.
    e. **Documentar los nuevos ficheros fuente y actualizar todos los modificados**. Actualizar la documentación técnica en HTML con ayuda de Doxygen.
    f. **Gestionar el proyecto a lo largo de la I4**, realizando reuniones (documentadas con actas que incluyan compromisos de los miembros del equipo, asignación de tareas, y plazos y condiciones de las entregas), **planificación** de la iteración del proyecto (tareas, recursos y tiempo, cronogramas mediante diagramas de Gantt), así como seguimiento de la planificación con ajustes, si fueran necesarios, que se reflejen en las actas y los cronogramas.

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 I4. 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, cuatro Bs y el resto Cs. Excepcionalmente sólo con tres Bs.
- **A**: Si se obtiene, al menos, cuatro As y el resto Bs. Excepcionalmente sólo con tres As.

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 ha entregado en el momento establecido todo el material solicitado. y (b) Es posible compilar y enlazar los ficheros fuentes de forma automatizada utilizando el `Makefile` entregado, tanto el programa del juego como los programas de prueba de los módulos.	Además de lo anterior: (a) La compilación y el enlazado no producen errores ni avisos (warnings) utilizando la opción `-Wall`. y (b) El `Makefile` entregado permite gestionar los fuentes del proyecto (limpiar `.o` y ejecutables, generar ayuda de usuario…), además de compilar y enlazar.	Además de lo anterior: (a) La compilación y el enlazado no producen avisos utilizando las opciones `-Wall -ansi -pedantic`. y (b) El `Makefile` entregado permite la generación de la documentación técnica del proyecto utilizando Doxygen usando la tarea por defecto.
Funcionalidad	Se han cubierto los requisitos de R1 a R12, de forma que se consigue el correcto funcionamiento de la aventura creada.	Además de lo anterior: (a) Se han cubierto los requisitos R13 y R14. y (b) Se consigue el correcto funcionamiento de la aventura creada utilizando los nuevos módulos implementados.	Además de lo anterior: (a) Se han cubierto los requisitos R15, R16. y (b) Se consigue el correcto funcionamiento de la aventura creada utilizando los nuevos módulos implementados.
Pruebas	Se han realizado al menos dos pruebas unitarias relevantes para cada función de los módulos involucrados en los requisitos de R1 a R12.	Además de lo anterior: Se han realizado al menos dos pruebas unitarias relevantes para cada función de los módulos involucrados en los requisitos de R13 y R14.	Además de lo anterior: Se han realizado al menos dos pruebas unitarias relevantes para cada función de los módulos involucrados en los requisitos R15 y R16.

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, estructuras y tipos públicos se hayan comentado. y (c) Que el código esté bien indentado[1]. y (d) Que los ficheros y las funciones incluyan cabeceras, y tengan todos los campos requeridos y estén correctamente comentadas, incluyendo datos de autor único.	Además de lo anterior: (a) Que NO se violen las interfaces de los módulos. y (b) Que se hayan incluido comentarios de Doxygen en: • Cabeceras de TODOS los ficheros. • Prototipos de funciones. • Tipos enumerados y estructuras de datos. y (c) Que se controlen los argumentos pasados a las funciones y los retornos de las funciones de entrada y reserva de recursos.	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. y (c) Que se genere correctamente la documentación técnica del proyecto con Doxygen en formato HTML.
Gestión de proyecto	Que se entregue, al menos, un acta de reunión con un diagrama de Gantt donde se muestre y justifique de forma razonable la organización del trabajo en el equipo para la I4.	Además de lo anterior: Que se entreguen, al menos, dos actas de reunión con diagramas de Gantt que muestren y justifiquen de forma razonable la organización inicial y final del trabajo en el equipo a lo largo de la I4.	Además de lo anterior: Que se entreguen actas de reunión con diagramas de Gantt actualizados por cada semana de I4, en las que se muestre y justifique de forma razonable la evolución de la organización del trabajo en el equipo durante la iteración.

[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.