Cuarta Iteración del Proyecto (I4): Juego Conversacional (Producto Final)

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, se añadirán otras que se consideren adecuadas para el Juego original que cada equipo deberá diseñar e implementar sobre el sistema.

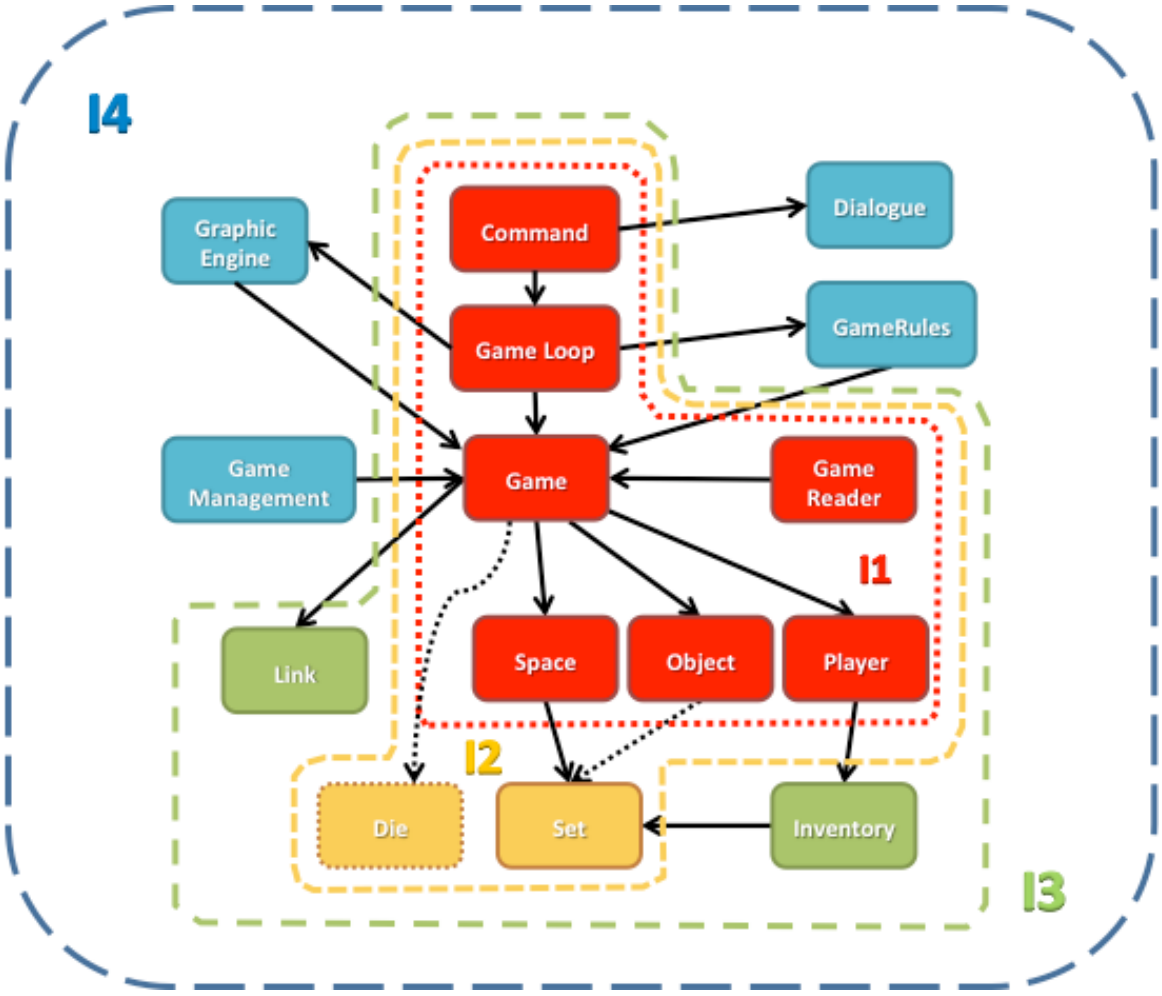

Figura 1. Módulos considerados en la cuarta iteración (I4) y final del desarrollo del proyecto.

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.

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 podrían desarrollarse 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 cuatro. El primero, *GameManagement* (GestorJuego) tiene la finalidad de dotar a la plataforma de la capacidad de gestionar partidas. El segundo, *GraphicEngine* (MotorGráfico) orientado a proporcionar una presentación mejorada del estado del juego. El tercero, *Dialogue* (Diálogo) destinado a dar al sistema de una interfaz más amigable y natural. Por último, el cuarto, *GameRules* (ReglasJuego) 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 provoquen. De todos estos módulos se hablará con más detalle a lo largo de este documento y en las clases de la asignatura.

El material de partida, resultado de I1, I2 e I3, debería generar aplicaciones que permitieran:

1. Cargar espacios, enlaces, objetos y jugadores desde ficheros 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 LOG 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 sus correspondientes ficheros de datos (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 (como los ".ent" proporcionados para probar el programa *ProtJuegoConv* de I3).

5. Programas (clientes 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.

Objetivos de la Cuarta Iteración y Final del Proyecto (I4)

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. 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, documentados mediante diagramas de Gantt), así como seguimiento de la planificación con ajustes, si fueran necesarios, que se reflejen en las actas y los cronogramas (diagramas de Gantt).

2. [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*).

3. [R2] Modificar también 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.

4. [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.

5. [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.

6. [R5] Completar el banco de pruebas para el módulo *Space* con las necesarias para las nuevas funcionalidades, modificar el programa de prueba del módulo (*space_test*) para que incorpore su implementación, y documentar el resultado de las mismas en el informe correspondiente.

7. [R6] Modificar el módulo *Object* para que incorpore soporte para nuevas propiedades junto con las funciones necesarias para su manipulación e impresión:

    a. Movible, 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 sólo podrá coger objetos para su inventario, si éstos son movibles.

    b. Movido, para señalar si el objeto movible se ha movido de su ubicación original. 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, para indicar si el objeto está oculto. Por omisión los objetos son visibles. En la descripción del espacio donde se encuentra el jugador sólo se dará información de los objetos visibles, aunque se puedan manipular normalmente (coger, dejar, examinar, etc.).

    d. Abre, 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, 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, para establecer si un objeto que puede iluminar está encendido o no. Se trata de un campo booleano, que por defecto tendrá el valor FALSO y que sólo se usará cuando el objeto puede iluminar.

8. [R7] Modificar el módulo *Object* otra vez para que incorpore, además de la descripción ya existente otra alternativa, para cuando el objeto no está en su

ubicación original. Implementar las funciones necesarias para manipular e imprimir dicho campo con los demás.

9. [R8] 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 sigue o no en su ubicación inicial, en caso de que el espacio esté iluminado.

10. [R9] Crear los comandos necesarios para encender y apagar objetos que 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.

11. [R10] Añadir también 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.

12. [R11] Completar el banco de pruebas para el módulo *Object* con las necesarias para las nuevas funcionalidades, modificar el programa de prueba del módulo (*object_test*) para que incorpore su implementación, y documentar el resultado de las mismas en el informe oportunos.

13. [R12] Modificar, en caso de necesidad, los ficheros de datos correspondientes 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.

14. [R13] Crear una aventura que incluya 10 espacios y 20 objetos, por lo menos, con su argumento y sus correspondientes ficheros de datos (espacios, enlaces, objetos, jugador, etc.).

15. [R14] 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; y (d) un fichero con la lista de comandos para seguir el camino anterior, tal como los ".ent" proporcionados para probar el programa *ProtJuegoConv* de I3.

16. [R15] Documentar los nuevos ficheros fuente y actualizar todos modificados. Actualizar la documentación técnica en HTML con ayuda de Doxygen.

17. [R16] Completar pruebas y programas de pruebas de todos los módulos.

18. Modificar el Makefile del proyecto para automatizar la compilación y el enlazado del conjunto.

19. Depurar el código hasta conseguir su correcto funcionamiento de la aventura creada.

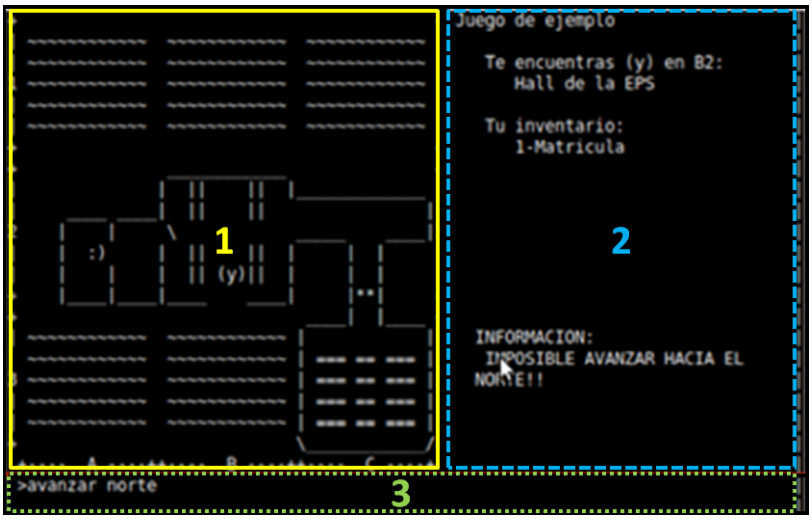

Figura 2. Interfaz básica del juego conversacional con tres zonas: (1) vista parcial del mapa del juego entorno al espacio donde se encuentra el jugador, (2) resto de información sobre el estado del juego, y (3) área para la entrada de instrucciones de usuario.

20. [R17] Crear un módulo *GraphicEngine* (MotorGráfico) que incorpore la funcionalidad necesaria para que se muestre por pantalla el estado del juego en cada momento y se permita la interacción del usuario. De esta forma, la interacción se realizará sobre una consola de 80x24 caracteres (80 columnas y 24 filas, ver Figura 2). La interfaz deberá incluir por lo menos tres zonas:

    a. Una primera donde se muestren el espacio donde está el jugador y todos los espacios conectados en torno a él (norte, noreste, este, sureste, sur, suroeste, oeste y noroeste), (zona 1 en la Figura 2). Por lo tanto, dicha zona debe tener hueco suficiente para las descripciones gráficas (ASCII) de 3x3 espacios (matriz de 9 posiciones). Las descripciones gráficas de los espacios en la I4 deberían ser por lo menos de 14 columnas por 7 filas, por lo que la zona tendrá en total 42x21 caracteres. En esta zona sólo se mostrarán las descripciones gráficas de los espacios que estén iluminados. En caso de que no haya espacio en alguna posición, se dejará vacío o se pondrá de manifiesto de algún modo (en la Figura 2 se indica con bloques de guiones).

    b. Una segunda zona donde se presentará el resto de la información del estado del juego (zona 2 en la Figura 2). Se deberá mostrar al menos información sobre: (a) la descripción del espacio donde se encuentra el jugador; (b) los nombres de los objetos visibles contenidos en dicho espacio; (c) los nombres de todos los objetos existentes en el inventario del jugador; y (d) un mensaje de retroalimentación sobre la última instrucción ejecutada por el usuario. La descripción del espacio y los objetos ubicados en él sólo se mostrarán si dicho espacio está iluminado o si el jugador lleva un objeto que ilumina encendido, en otro caso se indicara sólo que el espacio no está iluminado.

    c. Una tercera zona para que el usuario introduzca la instrucción a ejecutar (zona 3 en la Figura 2). Esta parte debe ocupar una o dos

líneas en la parte inferior de la consola.

21. [R18] Diseñar un banco de pruebas para el módulo *GraphicEngine*, crear un programa (*graphicengine_test*) que implemente las pruebas, realizar las mismas y documentar el proceso según se indicó en clase.

22. [R19] 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 un módulo que:

    a. Para cada comando ejecutado por el usuario, se acabe mostrando una frase indicativa de qué acción ha realizado y si esta se ha llevado a cabo con éxito o no. Por ejemplo, si el usuario invoca el comando *"go west"* y la ejecución ha sido correcta, el juego deberá construir y mostrar una frase del tipo *"You've gone 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, producirá un mensaje del tipo *"You can not go 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."*.

    Los alumnos podrán añadir todas las reglas de diálogos que 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.

23. [R20] Diseñar un banco de pruebas para el módulo *Dialogue*, crear un programa (*dialogue_test*) que implemente las pruebas, realizar las mismas y documentar el proceso se según se indicó en clase.

24. [R21] Implementación de un módulo *GameManagement* (GestorPartida). 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_save*) y otra para cargar (*game_load*). La función *game_save* almacenará en uno o varios ficheros (cuyo nombre deberá parametrizarse) el estado actual de la partida, esto es, el contenido de la estructura *Game*. Con respecto a *game_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/los fichero(s) cuyo nombre deberá(n) parametrizarse a la función**.**

25. [R22] Diseñar un banco de pruebas para el módulo *GameManagement*, crear un programa (*gamemanagement_test*) que implemente las pruebas, realizar

las mismas y documentar el proceso se según se indicó en clase.

26. [R23] 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 de fichero o ficheros donde guardarán los datos de la estructura *Game*. El comando para cargar una partida (*load*) deberá permitir indicar el nombre del fichero o ficheros de configuración con los datos que tiene actualmente la partida a cargar

27. [R24] Implementar un módulo *GameRules* (ReglasJuego). 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. Los alumnos podrán añadir tantas reglas de juego como consideren oportunas, con un mínimo de seis a su elección.

28. [R25] Diseñar un banco de pruebas para el módulo *GameRules*, crear un programa (*gamerules_test*) que implemente las pruebas, realizar las mismas y documentar el proceso se según se indicó en clase.

29. Modificar, en caso de necesidad, todos los módulos afectados por los cambios realizados, para mantener la funcionalidad previa e incorporar la nueva pretendida.

30. Volver a modificar el Makefile del proyecto considerando los nuevos módulos implementados, con el fin de mantener la automatización la compilación y el enlazado del proyecto completo.

31. Volver a depurar el código hasta conseguir su correcto funcionamiento con los módulos modificados y nuevos implementados.

Criterio 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 (40%). En particular, la calificación de este entregable se calculará según los siguientes criterios:

D: Si no se obtiene C en alguna de las columnas de la siguiente tabla de rúbrica.

C: Si se obtiene C en todas las columnas.

B: Si se obtienen, al menos, tres Bs y el resto Cs. Excepcionalmente sólo con dos Bs.

A: Si se obtiene, al menos, tres As y el resto Bs. Excepcionalmente sólo con dos As.

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, enlazar los ficheros fuentes del proyecto de forma automatizada utilizando el Makefile entregado, tanto el programa del juego como los programas de prueba de los módulos.	(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 TGZ con fuentes, datos y Makefile, ayuda de uso…).	(a) La compilación y el enlazado no producen avisos utilizando las opciones `-Wall -pedantic.` y (b) El Makefile entregado permite la generación de la documentación técnica del proyecto utilizando Doxygen.
Funcionalidad	Se han cubierto los requisitos de R1 a R16, de forma que se consigue el correcto funcionamiento de la aventura creada.	(a) Se han cubierto las parejas de requisitos, la R17-18 y la R19-20, o una de estas dos parejas y otros requisitos alternativos acordados con el profesor. y (b) Se consigue el correcto funcionamiento de la aventura creada utilizando los nuevos módulos implementados.	(a) Se han cubierto los requisitos R21-23, R24 -25, u otros requisitos equivalentes acordado con el profesor. 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 por cada función de todos los módulos incluidos.	Se han realizado al menos dos pruebas unitarias relevantes para cada función de todos los nuevos módulos implementados.	Se han realizado al menos dos pruebas unitarias relevantes para cada función de todos los nuevos módulos.
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. y (d) Que los ficheros fuente y las funciones incluyan cabeceras, y tengan todos los campos requeridos correctamente comentados, incluyendo los datos del autor único.	(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. • Cabeceras de funciones de ".h". • Tipos enumerados y estructuras de datos de ".h". y (c) Que se controlen los argumentos pasados a las funciones y los retornos de las funciones de entrada y reserva de recursos.	(a) Que el estilo sea homogéneo en todo el código. y (b) Que las variables locales de módulos o funciones que lo precisen se hayan comentado. y (c) Que NO se hayan incluido etiquetas de Doxygen en comentarios de tipos, estructuras ni cabeceras de funciones de ".c" y (d) Que se genere correctamente la documentación técnica del proyecto con Doxygen en formato HTML.
Gestión 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.	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.	Que se entreguen actas de reunión con diagramas de Gantt actualizados por cada semana de I4, que justifiquen la evolución de la organización del trabajo en el equipo durante la iteración razonablemente.