Tercera Iteración del Proyecto (I3): Prototipo de Juego Conversacional

En la tercera iteración (I3) se continúa con el desarrollo del proyecto y de las habilidades y los conceptos necesarios para ello, así como con la introducción y uso de herramientas apropiadas para dicha actividad. En esta iteración se abandonará el Juego de la Oca como referencia para el desarrollo, puesto que deja de ser un modelo adecuado para ilustrar las funcionalidades adicionales necesarias para completar un sistema que soporte Aventuras Conversacionales. Sin embargo, se demostrará la versatilidad del sistema conseguido, creando una nueva edición del Juego de la Oca que utilice las nuevas versiones de los módulos del proyecto obtenidas como resultado de esta iteración.

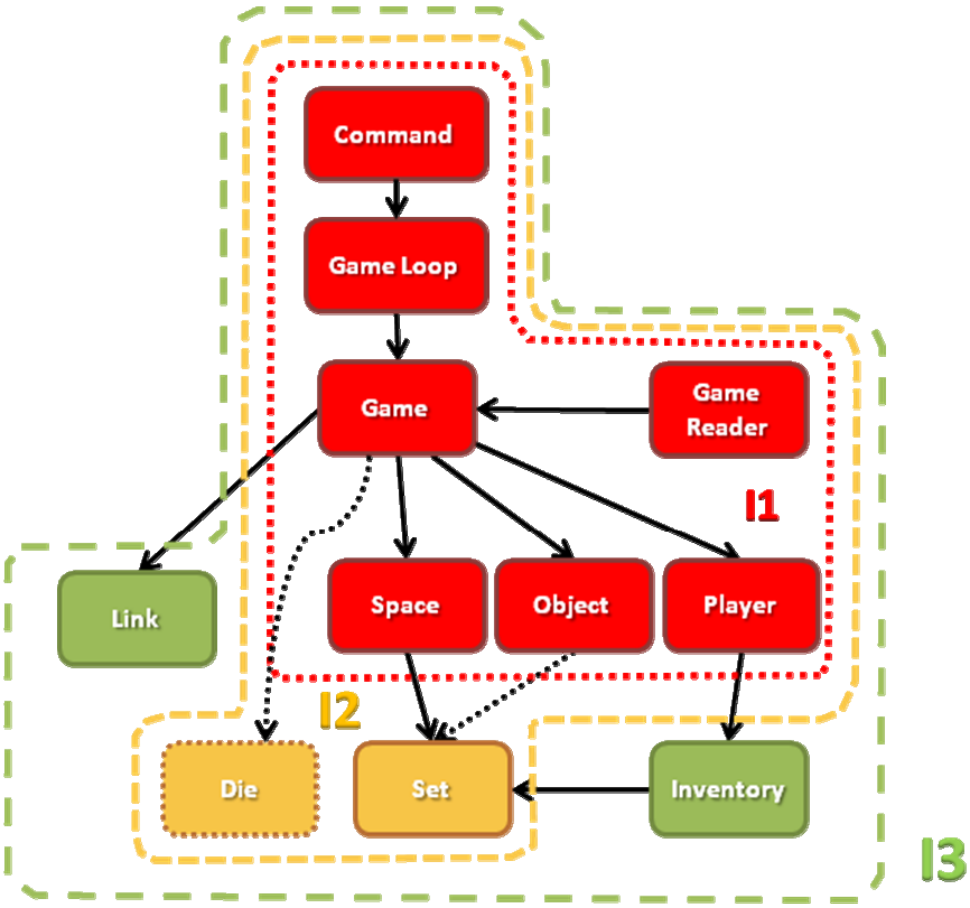

Figura 1. Módulos considerados en la tercera iteración (I3) del desarrollo proyecto.

La Figura 1 ilustra los módulos del proyecto en los que se trabajará en la I3, partiendo del material elaborado en las anteriores iteraciones (I1 e I2). En esta ocasión se completará el desarrollo de los módulos fundamentales del sistema como se explica en el documento de introducción al 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. Finalmente, en verde se muestran los módulos que se desarrollarán desde cero en la I3, en la que también se ampliarán y utilizarán algunos de los módulos de colores rojo y ocre. Los

nuevos módulos son dos *Inventory* (Inventario) y *Link* (Enlace). El módulo *Inventory* aportará la funcionalidad necesaria para dotar a cada jugador de un inventario (mochila) con el que llevar el conjunto de objetos que porte. El módulo *Link* proporcionará la funcionalidad para dotar a las conexiones entre espacios de capacidades nuevas, como por ejemplo estar abiertas o cerradas, o necesitar de un objeto (llave) para poderlas usar, como pasa con algunas puertas.

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

1. Cargar espacios y objetos desde ficheros de datos.
2. Gestionar todo lo necesario para la implementación de juegos con las características del de la Oca (espacios, objetos, jugador capaz de llevar un objeto, posicionamiento en el tablero de un objeto y del jugador, así como movimiento entre espacios enlazados entre sí).
3. Soportar la interacción del usuario con el sistema, interpretando comandos para mover al jugador, manipular objetos (el jugador sólo puede portar un objeto en I2, pero puede haber varios de ellos en cada espacio) y salir del programa (además de utilizar un dado).
4. Mostrar la posición del jugador en cada momento y las descripciones gráficas (ASCII) del espacio en el que se encuentra y de los contiguos al mismo, además de la ubicación de los objetos y el último valor del dado.
5. Liberar todos los recursos utilizados antes de terminar la ejecución del programa.

Como resultado de la I3, se esperan dos programas:

1. Una aplicación que permita la prueba de mapas formados por espacios unidos con enlaces que pueden estar abiertos o cerrados, objetos que tienen nombres para manipularlos y descripciones textuales que pueden consultarse, así como espacios con descripciones textuales (no sólo gráficas ASCII) que pueden consultarse también y un jugador que puede llevar más de un objeto. La finalidad de este programa es demostrar las nuevas funcionalidades incorporadas que ya son más de las necesarias para implementar el Juego de la Oca.
2. Al mismo tiempo, que la misma aplicación siga cubriendo las funcionalidades del Juego de la Oca, pero utilizando espacios conectados mediante enlaces (links) siempre abiertos, comandos para moverse en las direcciones que estén disponibles de las cuatro existentes (norte, sur, este y oeste) como alternativa a los movimientos ya implementados (adelante, atrás y salto), además de un jugador capaz de llevar varias fichas (objetos), aunque sólo llegue a portar una ficha jugando a la Oca (se puede conseguir limitando la capacidad del inventario utilizado), y un comando para dejar un objeto concreto del inventario del usuario (si cupiera más de uno en él). El objetivo de este programa es que demuestre la versatilidad del sistema implementado hasta el momento.

Todo ello incorporando previamente en los módulos del sistema las siguientes funcionalidades:

1. Cargar los enlaces entre espacios desde ficheros de datos, como se hace con los espacios y los objetos.
2. Gestionar todos los datos necesarios (espacios, enlaces, objetos, jugador y, en su caso, dado) para la implementación de las aplicaciones con las características indicadas.
3. Soportar la interacción del usuario para llevar varios objetos, examinar las descripciones de espacios y objetos, y moverse entre espacios conectados mediante enlaces en cuatro direcciones (norte, sur, este y oeste).
4. Mostrar las descripciones existentes de espacios y objetos.

Objetivos de la Tercera Iteración del Proyecto (I3)

Los objetivos de esta tercera iteración del proyecto (I3) son de dos tipos como en iteraciones anteriores. Por un lado, dirigidos al desarrollo de conocimiento y habilidades, en concreto a iniciarse en la gestión de proyectos y en la automatización de pruebas y de documentación (*Doxygen*), así como profundizar en el uso de herramientas para la depuración de programas (*gdb y valgrind*). Por otro lado, poner en práctica todo lo aprendido, en particular modificando el material obtenido de la I1 y la I2 para mejorarlo y dotarlo de nuevas funcionalidades, además de implementar aplicaciones nuevas que prueben y demuestren las funcionalidades incorporadas.

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 I3, 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 (Gantts).

2. [R1] Modificar el módulo *Game* para hacer opaca su estructura de datos, como se hizo en *Space* y debería hacerse en el resto de módulos del proyecto.

3. [R2] Crear un módulo *Inventory* (inventario) que incorpore la funcionalidad necesaria para dotar a cada jugador de un contenedor (mochila) donde llevar el identificador de varios objetos, de forma parecida a como lo hace el módulo *Space* para los espacios. En particular, los inventarios deberán implementarse como una estructura de datos con dos campos, uno para almacenar el conjunto de identificadores (utilizando el módulo *Set* desarrollado en la I2), y otro para establecer el número máximo de objetos que el usuario puede llegar a portar (este número máximo es independiente del tamaño máximo definido en la implementación de *Set*, ya que representa el tamaño de la mochila). Además, como otras veces, deberán implementarse las funciones necesarias para crear y destruir el inventario (*create* y *destroy*),

cambiar los valores de sus campos (como *set* y *get*) e imprimir el contenido de los mismos para su depuración (*print*).

4. [R3] Siguiendo las indicaciones y especificaciones que se proporcionen en clase sobre automatización de pruebas, diseñar un banco de pruebas para el módulo *Inventory*, crear un programa (*inventory_test*) que las implemente y documentar el resultado de las mismas en un informe.

5. [R4] Modificar el módulo *Player* para que, utilizando un inventario (mediante el módulo *Inventory*), permita que los jugadores puedan llevar varios objetos consigo. Para ello debería sustituirse en la estructura de datos de *Player* el identificador de objeto portado, por un campo *Inventory* (que permite acceder a la descripción del máximo de objetos permitidos y el conjunto de los mismos). Además de la estructura de datos de *Player*, deberán modificarse todas las primitivas que utilicen el antiguo campo sustituido, para que sigan funcionando con el nuevo, así como añadir funciones adicionales, en caso de necesidad, para manejar correctamente el inventario. Por ejemplo, funciones para añadir un elemento a la mochila del jugador, para obtener los identificadores de objetos en ella, o para saber si el identificador de un objeto está en la misma.

6. [R5] Diseñar un banco de pruebas para el módulo *Player*, crear un programa (*player_test*) o modificar el existente para que las implemente. Documentar el resultado de las mismas en un informe con el formato. Todo ello según se especifique en clase.

7. [R6] Crear un módulo *Link* (enlace), siguiendo también el modelo de *Space*, que proporcione la funcionalidad necesaria para el manejo de enlaces entre espacios. En concreto, los enlaces deberán implementarse como una estructura de datos con varios campos: un identificador único para cada enlace, un nombre, un par de campos con identificadores para indicar los espacios enlazados entre sí, y un campo para determinar el estado del enlace (abierto o cerrado). Como siempre, el módulo debería facilitar las funciones necesarias para crear y destruir enlaces (*create* y *destroy*), cambiar los valores de sus campos (como *set* y *get*) e imprimir el contenido de los mismos para su depuración (*print*).

8. [R7] Diseñar un banco de pruebas para el módulo *Link*, crear un programa (*link_test*) que las implemente. Documentar el resultado de las mismas en un informe con el formato. Todo ello según se especifique en clase.

9. [R8] Modificar el módulo *Space* para que utilizando enlaces (mediante el módulo *Link*), permita conectar los espacios de forma más potente. Para ello deberían sustituirse en la estructura de datos de *Space* los identificadores de espacios conectados al norte, este, sur y oeste, por los identificadores de los enlaces (*links*) que establecen las correspondientes conexiones con los espacios en los cuatro puntos cardinales. Además de la estructura de datos de *Space* deberán modificarse todas las primitivas que utilicen los campos mencionados, así como añadir funciones nuevas, en caso de necesidad para manejar correctamente los enlaces. Por ejemplo, funciones para determinar

si un espacio está unido con otro a través de un enlace o si un enlace es practicable (porque está abierto).

10. [R9] Diseñar un banco de pruebas para el módulo *Space*, crear un programa (*space_test*) o modificar el existente para que las implemente. Documentar el resultado de las mismas en un informe con el formato. Todo ello según se especifique en clase.

11. [R10] Crear un fichero de carga de enlaces (*links*), siguiendo el modelo utilizado originalmente para los espacios.

12. [R11] Crear también una función en *GameReader* para cargar los enlaces en el juego similar a la utilizada para cargar los espacios y los objetos.

13. [R12] Añadir un nuevo comando para moverse por los espacios siguiendo los enlaces de forma genérica, de manera que se especifique el comando seguido de la dirección del movimiento. Por ejemplo *"go north"* o *"go west"* (o de forma compacta "g n" o "g w"), de manera parecida a como se hacía en I2 para coger un determinado objeto.

14. [R13] Añadir otro comando para examinar objetos, obteniendo la descripción de los mismos. Para ello deberá indicarse el objeto que se quiere examinar. Por ejemplo *"inspect O1"* (o de forma compacta *"i O1"*), tal como se propuso en casos anteriores. Sólo se podrán examinar objetos que estén en el espacio donde está el jugador o que este tenga en su mochila (inventario).

15. [R14] Modificar el comando de inspeccionar objetos para que también se pueda inspeccionar el espacio donde está el usuario, utilizando para ello un término especial reservado (*"space"*). Por ejemplo *"inspect space"* (o de forma compacta *"i s"*). El término especial no podrá utilizarse para referirse a objetos.

16. [R15] Modificar, en caso de necesidad, los demás módulos existentes para que utilicen los nuevos implementados y los modificados, para mantener la funcionalidad previa e incorporar la nueva pretendida. Por ejemplo, en el módulo *Game*: (a) incluir un array en la estructura de datos correspondiente para guardar todos los enlaces establecidos, como se hace con los espacios y los objetos; (b) modificar las funciones de inicialización para ocuparse del nuevo campo correctamente; y (c) modificar todas las funciones existentes que deban utilizar dicho campo. Modificar los programas de prueba existentes de cada módulo para probar los cambios realizados.

17. [R16] Añadir un parámetro –nv (*no verbose*) que haga que *GameLoop* no muestre nada por consola (deshabilite las llamadas a imprimir por pantalla) para facilitar las pruebas de las nuevas funcionalidades incorporadas al programa utilizando ficheros de instrucciones en lugar de la entrada interactiva por teclado. Con esto podrá recibir instrucciones desde archivos de texto que contengan un comando por cada línea con el formato:

VERBO OBJECTG

Donde *VERBO* debe ser el nombre de una acción, en particular "*go*" (ir), "*take*" (coger), "*leave*" (dejar), "*inspect*" (examinar) y "*quit*" (salir). Y donde *OBJECTG* debe ser el objeto gramatical sobre el que recae la acción, como el nombre de algún punto cardinal, en concreto "north", "east", "south" y "west", el nombre de algún objeto del juego previamente cargado (como podrían ser "*lantern*", "*notebook*", o "*gun*", que son linterna, cuaderno y pistola respectivamente), alguna palabra reservada, en particular "*space*" de la que se habla en un apartado anterior, o nada, para las acciones que no necesitan objeto gramatical, que es el caso de "*quit*". Es imprescindible ceñirse a esta nomenclatura. Un ejemplo de este tipo de archivo podría ser:

```
go north
go north
go west
take lantern
go west
leave lantern
inspect lantern
…
quit
```

Para recibir las instrucciones del archivo se utilizara la redirección de la entrada de los comandos que proporcionan los intérpretes de comandos de la terminal. De este modo las instrucciones que el usuario debería ir escribiendo de forma interactiva, el programa las irá leyendo directamente de un fichero preparado previamente (al estilo del ejemplo anterior). Por lo tanto, no será necesario cambiar el programa, porque a todos los efectos el fichero se convertirá en la entrada estándar (que de otro modo y habitualmente es el teclado del ordenador). Por ejemplo, si para jugar al Juego de la Oca de la I2 el comando para invocar el programa es "*./juegoca datos1*", si preparamos un fichero con instrucciones de juego "partida1.oca", la línea de comando "*./juegoca datos1 < partida1.oca*" irá leyendo las instrucciones del fichero como si las fuera escribiendo el usuario. En clase se presentará un ejemplo práctico de este tipo de redirección.

Para cada instrucción recibida del archivo de texto, el programa tendrá que ejecutarla, modificando el estado del juego cuando sea necesario, y escribir en un fichero de LOG la instrucción ejecutada junto con el resultado de la misma. A modo de ejemplo, el programa podría generar el siguiente fichero de LOG en respuesta a la secuencia de comandos del ejemplo anterior:

```
go north: OK
go north: ERROR
go west: OK
take lantern: OK
go west: ERROR
leave lantern: OK
inspect lantern: OK
…
quit: OK
```

El programa seguirá ejecutando la secuencia de instrucciones del fichero hasta que se llegue al final del mismo, aunque algunas instrucciones no terminen bien.

18. [R17] Adaptar la aplicación anterior para que se pueda utilizar con los lotes de ficheros de datos que se proporcionan (lotes formados por ficheros de carga para espacios, enlaces, objetos y jugadores) junto con otro de las correspondientes salidas previstas. Probar y depurar la aplicación hasta conseguir que con ella se obtengan las salidas esperadas para cada lote proporcionado.

19. Documentar el proyecto utilizando Doxygen. Primero, incluyendo en todos los ficheros fuentes etiquetas adecuadas (según se indique en clase), en concreto: (a) en los comentarios de cabecera de TODOS los ".h" y ".c"; (b) en los de cabecera de funciones de los ".h"; y (c) en los comentarios de tipos enumerados y estructuras de datos de los ".h". Después, generando la documentación con ayuda de la utilidad, por lo menos en formato HTML.

20. [R18] Modificar la aplicación del Juego de la Oca resultado de la I2 para que siga funcionado como lo hacía, pero utilizando las nuevas versiones de los módulos del sistema (JuegoOcaPlus). Hacer lo necesario para que, además, se puedan emplear los siguientes comandos en sustitución de algunos comandos primitivos (o los equivalentes en la implementación de cada equipo): *"go south"* como alternativa a *"next"*, *"go north"* alternativa a *"back"*, y *"go east"* o *"go west"* como alternativa a *"jump"*. Hacer también que la visualización del estado del juego muestre los enlaces de las casillas visibles con sus correspondientes identificadores y los espacios destino de los mismos.

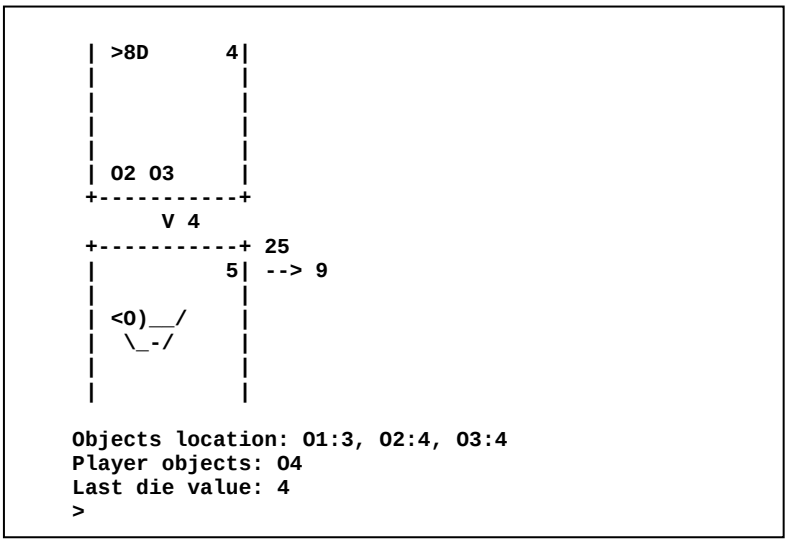

Figura 2. *Fragmento de visualización de estado de juego: arriba casilla **4** con jugador "**>8D**" y fichas "**O2**" y "**O3**"; debajo, enlace **4**, casilla de oca **5** y al este de ella enlace **25** a la casilla **9**; y en la parte inferior, localización de objetos, objeto portado por el jugador, último valor del dado y el prompt del sistema.*

21. [R19] Crear y modificar los ficheros de datos necesarios para la nueva versión del Juego de la Oca (JuegoOcaPlus). En concreto, crear el fichero de enlaces y modificar el de espacios adecuadamente. Para facilitar esta tarea se puede partir de un mapa del tablero de la Oca como el de la Figura 3:

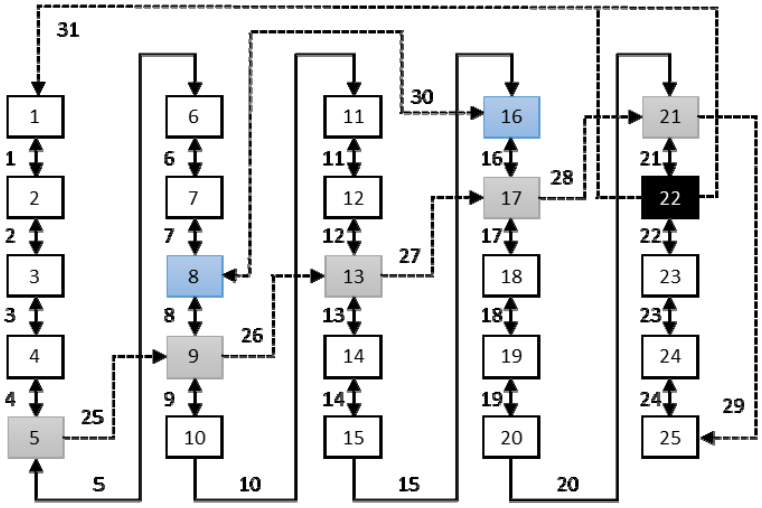

Figura 3. *Ejemplos de mapa del tablero de la oca, en el que se representan los espacios (casillas) con cajas y los enlaces entre ellos con flechas, ambas cosas etiquetadas con sus identificadores numéricos (que se asignarán en los ficheros de datos).*

En la Figura 3, las casillas de ocas (5, 9, 13, 17 y 21) se representan con fondo gris, las de puentes (8 y 16) con fondo azul y la de muerte (22) con fondo negro. Los enlaces norte/sur (secuenciales en la oca) se muestran con flechas de línea continua y los este/oeste (especiales en algunas casillas, como las de oca, puente y muerte) con flechas de línea descontinua. Las flechas dobles indican enlaces que se utilizan para comunicar espacios en ambos sentidos, por ejemplo la casilla 2 se comunica por el sur con la 3 y la 3 se comunica por el norte con la 2, en ambos casos mediante el enlace 2. Las flechas simples indican conexiones en un solo sentido, por ejemplo la casilla 13 se comunica por el este con la casilla 17 mediante el enlace 27, pero la 17 no lo hace con la 13 por el este. La casilla 22 (muerte) está comunicada por el este y el oeste con la 1 mediante el enlace 31.

Tabla 1. *Ficheros de datos relativos a espacios al estilo de la I2 y de la I3*

I2: Fragmento de fichero de datos de espacios al estilo de la **I2**, donde las conexiones entre espacios se establecen directamente indicado identificadores de espacios definidos en el mismo fichero.	Fragmento fichero datos espacios para I2 … #s:3|Spc 3|2|-1|4|-1| #s:4|Spc 4|3|-1|5|-1| #s:5|Spc 5 (Oca)|4|-1|6|9| |<o)__/ | _=/ | #s:6|Spc 6|5|-1|7|-1| #s:7|Spc 7|6|-1|8|-1| …
I3: Fragmentos de ficheros de datos de espacios (derecha arriba) y de enlaces (derecha abajo) al estilo de I3, donde las conexiones entre espacios se establecen mediante enlaces definidos en un fichero de datos de enlaces. Los enlaces pueden tener propiedades particulares (como abierto-cerrado) que en la implementación para I2 no eran posible. En concreto en el fichero de enlaces la columna 1 establece el id del enlace, la 2 el id de uno de los espacios enlazados y la 3 el del otro, y la 4 indica si el enlace está abiertos o cerrados (0 o 1 respectivamente).	Fragmento fichero datos espacios para I3 … #s:3|Casilla 3|2|-1|3|-1| #s:4|Casilla 4|3|-1|4|-1| #s:5|Casilla 5 (Oca)|4|-1|5|25| |<o)__/ | _=/ | #s:6|Casilla 6|5|-1|6|-1| #s:7|Casilla 7|6|-1|7|-1| … Fragmento fichero datos enlaces para I3 … #l:2|Lnk 2|2|3|0| #l:3|Lnk 3|3|4|0| #l:4|Lnk 4|4|5|0| #l:5|Lnk 5|5|6|0| #l:6|Lnk 6|6|7|0| #l:7|Lnk 7|7|8|0| … #l:25|Lnk 25|5|9|0| …

En la Tabla 1 se muestran un ejemplo de fragmento de fichero de datos de espacios como se implementó en la I2 (fila I2) y la implementación equivalente del fragmento para la I3 utilizando un fichero de espacios y otro de enlaces (filas I3) de acuerdo todo ello con el mapa de la Figura 3.

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

23. Depurar el código hasta conseguir su correcto funcionamiento.

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 I3 (30%). 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.

Tabla de rúbrica

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 de forma automatizada utilizando el Makefile entregado.	(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…), además de compilar y enlazar.	(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 R9.	Se han cubierto los requisitos de R10 a R17.	Se han cubierto los requisitos R18 y R19.
Pruebas	Se han realizado al menos dos pruebas unitarias relevantes (tal como se indique en clase) para cada función de los módulos involucrados en los requisitos de R1 a R9.	Se superan más de la mitad de los casos de prueba proporcionados para ProtoJuegoConv.	(a) Se superan todos los casos de prueba proporcionados para ProtoJuegoConv. O (b) JuegoOcaPlus mantiene la funcionalidad de I2 con R19 y R20
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 fuentes y las funciones incluyan cabeceras, y tengan todos los campos requeridos y estén correctamente comentadas, 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 a cada módulo o función que precisen explicación se hayan comentado. y (c) Que NO se hayan incluido comentarios de Doxygen en 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 I3.	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 I3.	Que se entreguen actas de reunión con diagramas de Gantt actualizados por cada semana de I3, en las que se muestre y justifique de forma razonable la evolución de la organización del trabajo en el equipo durante la iteración.