

I2: Objetos y personajes

Introducción

Se va a comenzar el desarrollo de la segunda iteración del proyecto (I2) añadiendo nueva funcionalidad al motor de juego conversacional iniciado. La Figura 1 ilustra los módulos del proyecto en los que se trabajará en esta nueva iteración a partir del material elaborado en la primera (I1). Los módulos obtenidos como resultado de la I1, y que se utilizarán y ampliarán en esta nueva entrega, se han representado en **rojo**; mientras los módulos que se desarrollarán en la segunda iteración se presentan en **amarillo**.

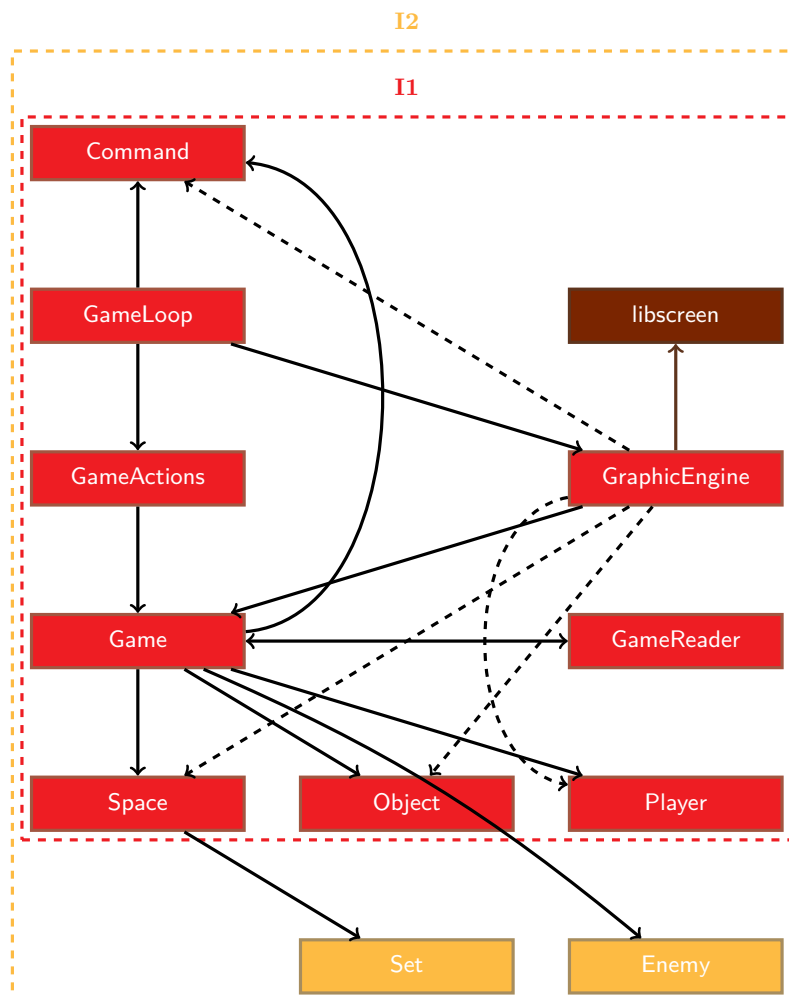


Figura 1: Módulos considerados en la segunda iteración (I2) del desarrollo del proyecto.

Los nuevos módulos son dos: **Set** (Conjunto) y **Character** (Personaje). El módulo **Set** proporcionará la funcionalidad necesaria para manejar conjuntos de identificadores, como los que tienen el jugador, los objetos o los espacios. Un conjunto es una colección de elementos distintos (no repetidos) en la que no se considera el orden. El módulo **Character** es un módulo similar al módulo **Player** y permitirá introducir en el juego otros personajes (no jugadores) con los que interactuar.

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

1. Cargar los espacios de juego desde un fichero de datos.
2. Gestionar todo lo necesario para la implementación del juego (espacios, jugador capaz de llevar un objeto, y posicionamiento en el mapa de un objeto y del jugador).
3. Soportar la interacción del usuario con el sistema, interpretando comandos para mover al jugador por el mapa (adelante y atrás), hacer que el jugador manipule un objeto en los espacios (cogerlo y dejarlo), y salir del programa.
4. Mover al jugador por los espacios, pudiendo coger y dejar el objeto, y haciendo cambiar el estado del juego.
5. Mostrar la posición en cada momento del jugador y de las casillas contiguas (norte y sur) a la que ocupa, indicando también la ubicación del objeto.
6. Liberar todos los recursos utilizados antes de terminar la ejecución del programa.

Como resultado de la I2, se espera que la aplicación siga cubriendo las funcionalidades de la I1, pero que con los nuevos requisitos además permita:

1. Cargar los objetos involucrados en el juego desde un fichero de datos, como se hace con los espacios.
2. Gestionar todos los datos necesarios para la implementación del juego con las nuevas características que se establezcan.
3. Mejorar la interfaz gráfica, mostrando las cuatro conexiones de los espacios, la posición de todos los objetos, de los nuevos personajes y el resto de información relevante del juego.

Para esta nueva iteración se parte del mapa ampliado creado en la I1 y que se muestra en la Figura 2.

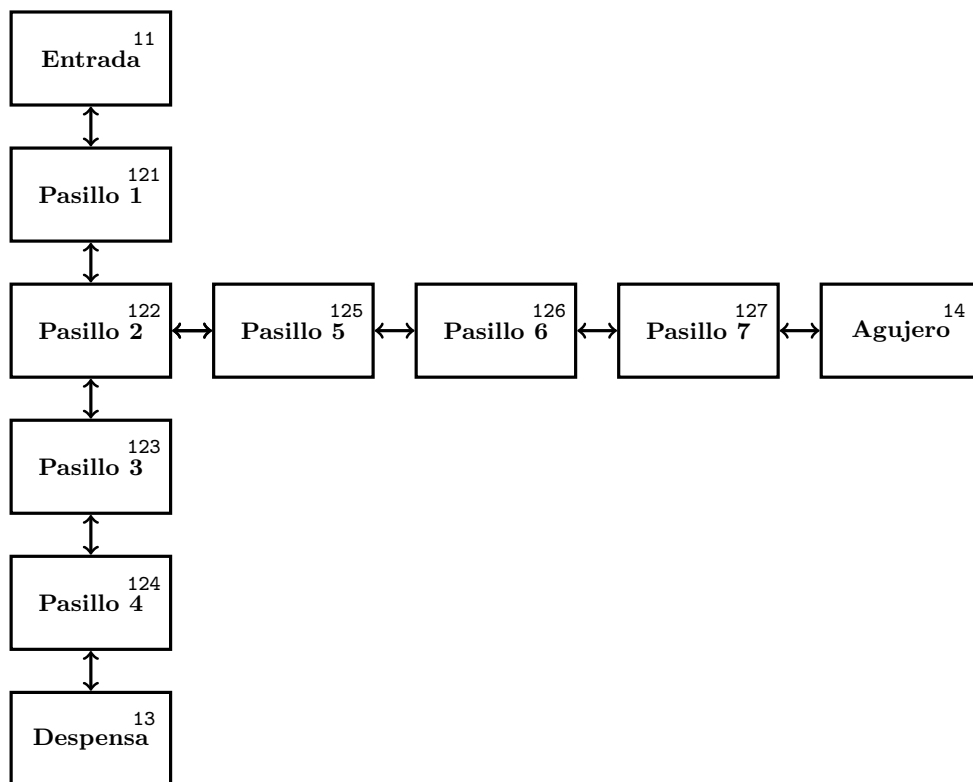


Figura 2: Mapa del hormiguero.

Objetivos

Los objetivos de esta segunda iteración del proyecto son de dos tipos. Por un lado, profundizar en el empleo del entorno de programación de GNU (`gcc`, `gdb`, etc.), así como iniciarse en la depuración de programas a través de herramientas como Valgrind o con la creación de trazas y pruebas de los módulos del proyecto. Se debe revisar para todo ello el módulo “Gestión de errores de programación” del curso [SPOC](#) así como los materiales disponibles en [Moodle](#). Por otro lado, practicar todo ello con el material disponible como resultado de la I1, modificando 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 durante la primera semana de la iteración.
- 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 de forma automatizada utilizando el fichero `Makefile` entregado. Es fundamental actualizar este fichero a medida que se van añadiendo nuevos ficheros y nuevas dependencias.
- C2.** Comprobar que no se producen *warnings* al compilar con la opción `-Wall -ansi -pedantic`.

Requisitos de documentación y estilo

- D1.** Documentar **todos** los ficheros del proyecto. En concreto se debe intentar:
 - 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 indentado¹.
 - Que el estilo sea homogéneo en todo el código².

¹La 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.

²Como 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.** Modificar el módulo **Game** para hacer opaca su estructura de datos, como se ha hecho en el resto de módulos del proyecto.
- F2.** Crear un módulo **Set** (conjunto) que integre la funcionalidad necesaria para el manejo de conjuntos. En particular, los conjuntos deberán implementarse como una estructura de datos con dos campos, uno para almacenar un array de identificadores (**ids**), y otro para recordar el número de ellos en cada momento (**n_ids**). Este módulo debe facilitar al menos las funciones necesarias para crear y destruir conjuntos (**create** y **destroy**), añadir y eliminar valores (**add** y **del**) e imprimir el contenido de los mismos para su depuración (**print**). Para las funciones **add** y **del** se deberá entregar el pseudocódigo con el algoritmo diseñado para su ejecución a lo largo de la segunda semana del proyecto. En este módulo se pueden añadir otras funciones necesarias para el manejo de conjuntos desde fuera del TAD.
- F3.** Modificar el módulo **Space** para que, utilizando un conjunto (mediante el módulo **Set**), permita que los espacios puedan contener varios objetos en su interior. Para ello debería sustituirse en la estructura de datos de **Space** el actual campo **object** por otro campo **objects** que contenga el conjunto de los objetos en el espacio. Además de la estructura de datos de **Space**, deberán modificarse todas las primitivas que utilicen el campo sustituido para que sigan funcionando con el nuevo. Se añadirán, además, todas aquellas funciones adicionales que se consideren necesarias para manejar correctamente los objetos. Por ejemplo, funciones para añadir un objeto al espacio, para acceder a los identificadores de objetos en dicho espacio, o para saber si el identificador de un objeto está en el mismo.
- F4.** Crear un módulo **Character** (Personaje) siguiendo el ejemplo del módulo **Player** (aunque su localización será parecida a la de los objetos). En concreto, deberá implementarse como una estructura de datos con campos de identificación (**id**), nombre (**name**), una cadena de 6 caracteres que contenga la descripción gráfica del personaje (**gdesc**, salud (**health**) para indicar el número de puntos de vida que posee en cada momento el personaje, un campo booleano que indique si el personaje es amigo (**friendly**) y una cadena de caracteres que almacene un mensaje (**message**). También en este caso, el módulo debería facilitar las funciones necesarias para crear y destruir personajes (**create** y **destroy**), leer y cambiar los valores de sus campos (**get** y **set**) e imprimir el contenido de los mismos (**print**).
- F5.** Añadir al módulo **Player** un campo que indique la salud del jugador (**health**) para almacenar el número de puntos de vida de los que dispone en cada momento.
- F6.** Modificar los demás módulos existentes para que utilicen los módulos anteriores manteniendo la funcionalidad previa. Por ejemplo, en la estructura de datos de **Game**, sustituir el puntero a **Object** existente para guardar el único objeto que se permitía en la I1 por un array de punteros a objeto para almacenar todos los que se utilicen en el juego, como se hace con los espacios. También se debe añadir a la estructura de **Game** un array de punteros para almacenar a los distintos personajes del juego. **Space** deberá ahora incluir un nuevo campo **character** que almacene el identificador del personaje que se encuentra en el espacio, o **NO_ID** si no hay ningún personaje. Cada espacio podrá contener como mucho un personaje. Además, se deben utilizar las primitivas adecuadas de los nuevos módulos para la manipulación necesaria de datos desde el resto de módulos.
- F7.** Añadir al fichero de carga de datos (**anthill.dat**) aquellos datos relativos a los objetos que se utilizarán en el juego, siguiendo el modelo empleado para cargar los espacios, con el formato **#o:21|Grain|11**, donde se indica el identificador del objeto, su nombre y su posición inicial. El fichero debe incluir cuatro objetos.
- F8.** Crear una función en **GameReader** para cargar los objetos en el juego, siguiendo el modelo de la función utilizada para cargar los espacios. Modificar los módulos necesarios para que

los objetos se carguen desde fichero, como se hace con los espacios.

- F9.** Añadir dos nuevos comandos (`left` o `l`, `right` o `r`) que permitan moverse por el mapa en dirección oeste y este respectivamente.
- F10.** Añadir un comando de ataque (`attack` o `a`) que permita a un jugador enfrentarse a un personaje si está en el mismo espacio y no es amigo. En este caso, se supone que los niveles del jugador y el adversario son equivalentes y, por tanto, cuando se ejecute este comando se generará un número aleatorio entre 0 y 9. Si el número está entre 0 y 4, gana el adversario y por tanto el jugador pierde un punto de vida. Si el número generado está entre 5 y 9, gana el jugador, y en este caso es el personaje el que pierde un punto de vida. Si el enemigo tiene 0 puntos de salud, está muerto y no se le puede atacar más. En el caso de que el jugador pierda todos los puntos de vida deberá marcarse el juego como finalizado. Es importante asegurarse de que el número generado es realmente aleatorio y no se genera siempre el mismo número o la misma secuencia de números.
- F11.** Añadir un comando de chat (`chat` o `c`) que permita a un jugador hablar con un personaje si está en el mismo espacio y es amigo. En este caso, se mostrará el mensaje del personaje en la descripción del juego hasta el siguiente comando.
- F12.** Modificar el comando `take` o `t` que permite al jugador coger un objeto de un espacio, de modo que pueda indicarse el objeto que se pretende coger de entre los que haya disponibles en el espacio concreto. En particular, este comando deberá incluir el nombre del objeto que se quiere coger, por ejemplo: `take grain`.
- F13.** Modificar una vez más el módulo `Space` para incluir una descripción gráfica (ASCII) del espacio de 5×9 caracteres. Para ello deberá incluirse un campo `gdesc` que sea un array de cinco strings, cada uno con espacio para nueve caracteres. De este modo, por ejemplo, la descripción gráfica del espacio con identificador 122 de la Figura 4 quedaría:

```

      _ _ _ _ _
     /   \   \   \
    /     \   \   \
   /       \   \   \
  /         \   \   \
 /           \   \   \
/             \   \   \
\             /   /   /
 \           /   /   /
  \         /   /   /
   \       /   /   /
    \     /   /   /
     \   /   /   /
      _ _ _ _ _

```

Deberán adaptarse las funciones existentes para manejar el nuevo campo (`create`, `destroy` y `print`) y crear las nuevas funciones de manipulación necesarias (`set` y `get`).

- F14.** Añadir al menos cuatro descripciones gráficas en el fichero de datos, como por ejemplo las indicadas en la Figura 3, en cuatro espacios diferentes, siguiendo el modelo dado en la Figura 4. Modificar la función de carga de espacios para adaptarse a los cambios introducidos con la descripción gráfica, quitando además información de los espacios con los que se relaciona. Por ejemplo, para la primera casilla la línea quedaría:

```
#s:11|Entry| _ | \ \ _ _ _ | \ mo' \ \ @ _ | \ _ / \ _ |
```

```

      _
     / \
    /  \
   /   \
  /    \
 /     \
/      \
\      /
 \    /
  \  /
   \ /
    /
   /
  /
 /
/
\
 \
  \
   \
    \
     \
      \

```

Figura 3: Ejemplos de representación gráfica de algunos espacios.

- F15.** Modificar la función de visualización del estado del juego para mostrar: (a) la descripción gráfica (ASCII) de cada espacio; (b) flechas que indiquen la dirección hacia la que puede moverse el jugador desde el espacio actual; (c) el espacio donde está cada objeto; (d) varios objetos en los espacios visualizados; (e) la descripción gráfica de los personajes en el espacio

```

-      +-----+                               - Objects:                                -
-      | /oo/\ 121|                             - Seed: 11                                  -
-      |          |                             - Crumb: 122                              -
-      |          |                             - Grain: 122                               -
-      |          |                             - Leaf: 123                               -
-      +-----+                               -                                     -
-                                         - Characters:                            -
-                                         - /\oo/\ : 121 (4)                       -
-                                         - ^Om  : 122 (5)                       -
-                                         -                                     -
-      +-----+ +-----+                     - Player : 122 (5)                         -
-      | m0^ ^Om 122| |         125|-        - Player has no object                    -
-      | _/_\_/ - | | _/_\_/ - |           - Message: Hi!                          -
-      | o o |   > | o o |               -                                     -
-      |mo' _ 0o8o | | mo' _ 0o8o |       -                                     -
-      |__/\_/\ | | __/\_/\_/\ |         -                                     -
-      |Crumb, Grain| |              |     -                                     -
-      +-----+ +-----+                     -                                     -
-                                         - v                                     -
-      +-----+                               -                                     -
-      |         123|                             -                                     -
-      | / / |                                   -                                     -
-      | / \__|                                 -                                     -
-      | mo'\ |                                 -                                     -
-      | _ @ _ \|                             -                                     -
-      | \_\_// / |                           -                                     -
-      |Leaf |                               -                                     -
-      +-----+                               -                                     -
-
-----
- The anthill game
-----
- The commands you can use are:
- next or n, back or b, left or l, right or r, take or t, drop or d, attack or a,
- chat or c, exit or e
-----
- Take (t): ERROR
- Chat (c): OK
- Next (n): OK

```

Requisitos de pruebas

- 6

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 I2. 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.

Tabla 1: Rúbrica para la segunda iteración (I2).

Objetivo	Puntuación	Aprobado	Notable	Sobresaliente
G1	0,50	<div></div>	<div></div>	<div></div>
G2	0,50	<div></div>	<div></div>	<div></div>
C1	1,50	<div></div>	<div></div>	<div></div>
C2	0,50	<div></div>	<div></div>	<div></div>
D1	1,00	<div></div>	<div></div>	<div></div>
D2	1,00	<div></div>	<div></div>	<div></div>
F1	0,30	<div></div>	<div></div>	<div></div>
F2	0,30	<div></div>	<div></div>	<div></div>
F3	0,30	<div></div>	<div></div>	<div></div>
F4	0,30	<div></div>	<div></div>	<div></div>
F5	0,10	<div></div>	<div></div>	<div></div>
F6	0,30	<div></div>	<div></div>	<div></div>
F7	0,20	<div></div>	<div></div>	<div></div>
F8	0,30	<div></div>	<div></div>	<div></div>
F9	0,30	<div></div>	<div></div>	<div></div>
F10	0,30	<div></div>	<div></div>	<div></div>
F11	0,30	<div></div>	<div></div>	<div></div>
F12	0,30	<div></div>	<div></div>	<div></div>
F13	0,20	<div></div>	<div></div>	<div></div>
F14	0,20	<div></div>	<div></div>	<div></div>
F15	0,30	<div></div>	<div></div>	<div></div>
P1	0,30	<div></div>	<div></div>	<div></div>
P2	0,35	<div></div>	<div></div>	<div></div>
P3	0,35	<div></div>	<div></div>	<div></div>

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

- No se ha completado el diagrama de Gantt.
- 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.
- No se ha implementado ningún requisito de pruebas.

Otras consideraciones

- Se penalizarán fuertemente los errores en tiempo de ejecución o cualquier otro error indicado por Valgrind.
- No se puntuarán aquellos requisitos de funcionalidad que no se puedan probar (o bien mediante un programa de prueba o bien mediante el propio juego).