

Práctica 04 v2 – Dungeon Master

Objetivo académico

Esta práctica consiste en terminar el juego Dungeon Master creado a propósito para que, por una parte puedas aplicar los conceptos teóricos que más abajo se requieren, y por otra todo aquello que hayas aprendido durante la asignatura. El fin de la práctica es que demuestres que eres capaz de resolver un problema antes no propuesto en los ejercicios/prácticas/clase.

El contexto: equipo de desarrolladores de juegos freelance

Formas parte de un equipo de desarrolladores de juegos freelance. Os ha llegado la propuesta de terminar un juego llamado Dungeon Master, con una parte de la codificación ya inicializada y un diseño básico predeterminado. Tenéis como fecha límite de entrega el 8 de Enero del 2018 ya que el cliente quiere distribuirlo en esas fechas, justo después de Reyes, momento en el que quiere sentarse con vosotros para hacer una revisión final.

Como sois un equipo sin miedo a enfrentaros a nuevos retos aceptáis la propuesta sin duda alguna.

Las funcionalidades: opciones del jugador

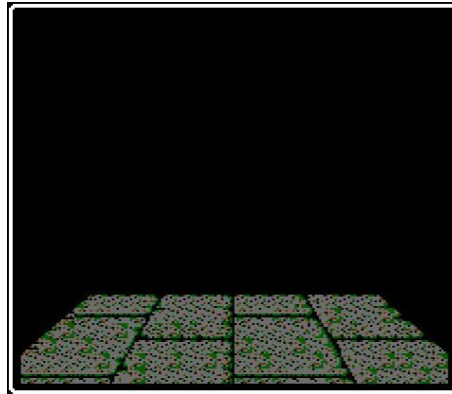
El juego debe contemplar las siguientes acciones:

- Moverse por la mazmorra: adelante, atrás, girar a la derecha y girar a la izquierda.
- Subir o bajar por los niveles de profundidad.
- Atacar y defenderse de posibles enemigos.
- Recoger y usar objetos.
- Un visor de navegación en el que visualizar la mazmorra.
- Un espacio donde ver el equipo y las características del jugador.
- Nueva, Guardar, Recuperar y Eliminar partida

El juego: Dungeon master

Dungeon Master es un juego de rol al más puro estilo dungeoncrawler 2D. La historia del jugador empieza en una mazmorra, desorientado y sin saber qué hace ahí. Su principal objetivo será mantenerse en vida y conseguir escapar de la mazmorra a toda costa.

La primera imagen que debe aparecer en el visor de navegación es la siguiente:



Nota para los desarrolladores: El jugador empieza en el nivel -2. Se pueden crear subniveles adicionales, pero principalmente el juego viene de serie con 2 mapas.

La mazmorra: niveles & mapa

La mazmorra consiste en un conjunto de mapas organizados en niveles. El nivel 0 se considera fuera de la mazmorra. El juego termina cuando el jugador llega al nivel 0.

El jugador debe poder moverse libremente por los mapas, aunque este debe estar delimitado por paredes, debe contener algún enemigo. Trampas, obstáculos u otras ocurrencias son opcionales.

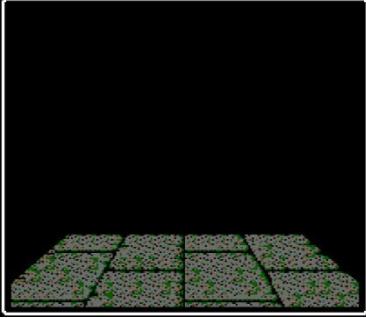
Nota para los desarrolladores: No es necesario mostrar el mapa del nivel. Esta opción queda a vuestra discreción así como la forma de mostrarlo y si en el mapa mostrar solo el recorrido descubierto o el mapa entero.

El mapa se declara como un array múltiple de 10x10 en el que internamente las paredes, objetos, enemigos, puertas, agujeros... se representan a conveniencia. Cada posición del array es una posición en el mapa.

El visor de navegación: lo que ve el jugador

La mazmorra es un lugar muy oscuro. A simple vista el jugador solo puede ver una posición por delante. Inicialmente el visor puede mostrar 2 posiciones distintas: oscuridad o entidad. Se considera una entidad todo aquello que no sea un espacio vacío, como puede ser un enemigo, una pared, una puerta o un objeto.

Ejemplos de situaciones según posición:

		
Sin pared delante	Con pared delante	Con enemigo delante
		
Con puerta delante para subir o bajar de nivel		

Nota para los desarrolladores: Puede modificarse la visión del jugador según modificador de raza, pócima, hechizo o cualquier ocurrencia. Una visión mejorada deberá reflejarse en el visor mostrando más posiciones por delante del jugador más allá de la inmediata. Es necesario definir el norte, sur, este y oeste a efectos de cargar una partida y mostrar al jugador la misma visión en el momento de guardar.

Para poder ir hacia la izquierda o a la derecha se deben utilizar las opciones de movimiento del jugador: girar a la izquierda y girar a la derecha. Esto significará que para ir a la izquierda el jugador deberá girar a la izquierda y avanzar.

El jugador: equipo y características

El jugador es de la raza humana y debe contemplar las siguientes características:

- **Nombre.**
- **Sexo.**
- **Nivel/XP:** Se empieza por nivel 1 y experiencia 0. Para superar el nivel se necesita $10 \times \text{nivel}$ puntos de experiencia acumulados. De esta forma un jugador de nivel 2 tendrá entre 20 y 49 puntos de experiencia. Un jugador de nivel 3 tendrá entre 50 y 89...
- **Ataque:** Se empieza por ataque +0. Por cada 2 niveles se sube 1 punto de ataque.
- **Defensa:** Se empieza por defensa +0. Por cada nivel se sube 1 punto de defensa.
- **Vida:** Se empieza por 10 puntos de vida. Por cada nivel se suben $10 \times \text{nivel}$ puntos de vida.

- **Objetos:** Se pueden tener tantos objetos como se quieran. El jugador tiene 2 manos y puede coger con ellas dos objetos, uno en cada mano.
 - Un objeto se describe como "nombre_de_objeto (+n ataque, +n defensa)"
 - Donde n es un número que indica un ataque positivo o negativo, o una defensa positiva o negativa.

Enemigos: fuente de experiencia

Los enemigos son una parte fundamental del juego para que el jugador pueda subir de experiencia. Un enemigo debe tener:

- Vida: Puntos de vida que el jugador deberá quitar para vencer.
- Ataque
- Defensa
- Puntos de experiencia: Puntos de experiencia que al morir se transferirán al jugador.
- Imagen: Imagen que se muestra al encontrarse con el enemigo.
- Objetos: Objetos que al morir se transferirán al jugador.

Nota para los desarrolladores: Se pueden crear otras situaciones que permitan subir de experiencia al jugador (resolver acertijo...).

La lucha: morir o sobrevivir

El sistema de lucha se basa en turnos y en las siguientes reglas:

- El jugador siempre ataca el primero.
- Por cada turno se restan ataque_atacante - defensa_atacado puntos de vida al atacado. Si la resta queda positiva significa que el atacado ha encajado el golpe y se le resta vida. Si la resta queda negativa significa que el atacado ha esquivado el golpe o el que ataca ha fallado el golpe.
- Cuando se vence a un enemigo sus objetos se añaden a los del jugador.
- Cuando se vence a un enemigo sus puntos de experiencia se suman a los del jugador.

Muerte: el jugador es vencido

Cualquier circunstancia que cause muerte al jugador significará que deberá recuperar una partida guardada o volver a empezar una partida nueva.

Escenario: diseño del juego

Nota para los desarrolladores: Inicialmente el diseño entregado es minimalista. Debe completarse con un diseño de juego de rol.

Partida: nueva, guardar, borrar o recuperar

Al cargar el juego debe mostrarse la opción de Nueva partida y, si hay alguna anteriormente guardada, Cargar partida.

En cualquier momento de la partida el jugador podrá guardar el estado de la misma. Solo pueden guardar dos partidas. Las partidas deben tener nombre. Si ya hay dos partidas guardadas el jugador deberá seleccionar qué partida eliminar.

Nota para los desarrolladores: Las partidas se guardan y recuperan en formato JSON (ver API).

API: comunicación con el servidor

La comunicación con el servidor se realizará con una API desarrollada a propósito para la práctica. Estas son las especificaciones de la API que el cliente indica:

- **POST:**
 - Descripción: Guarda una partida en el servidor.
 - Parámetros:
 - Token (en la url): identificador único de juego
 - Slot (en la url): nombre del slot en el que guardar la partida
 - Nueva: Guarda el JSON para nueva partida. El jugador no debe poder guardar en este slot. Solo debe usarse por los desarrolladores.
 - 1: Guarda el JSON en el slot 1
 - 2: Guarda el JSON en el slot 2
 - Json (en el paquete de datos): estado de la partida en formato JSON
 - Códigos de retorno:
 - 200 Si se ha guardado la partida correctamente
 - 404 Si ha ocurrido un error (por ejemplo el slot no está libre)
 - Contenido de retorno:
 - Ninguno
 - Formato de llamada:
 - `http://puigpedros.salleurl.edu/pwi/pac4/partida.php?token={token}&slot={nueva,1,2}`
- **GET:**
 - Descripción: Devuelve el contenido de una partida guardada en uno de los 2 slots disponibles, o una lista de los slots ocupados.

- Parámetros:
 - Token (en la url): identificador único de juego
 - Slot (en la url): nombre del slot. Si el nombre del slot no se indica se devuelven los slots ocupados.
- Códigos de retorno:
 - 200 Si se ha guardado la partida correctamente
 - 404 Si ha ocurrido un error (por ejemplo no existe la partida)
- Contenido de retorno:
 - JSON con el contenido de la partida del slot indicado o los slots con partida guardada.
- Formato de llamada para descargar configuración JSON:
 - `http://puigpedros.salleurl.edu/pwi/pac4/partida.php?token={token}&slot={nueva,1,2}`
- Formato de llamada para descargar lista de slots con partida guardada:
 - `http://puigpedros.salleurl.edu/pwi/pac4/partida.php?token={token}`
- **DELETE:**
 - Descripción: Elimina una partida guardada anteriormente.
 - Parámetros
 - token: identificador único de juego
 - slot: nombre del slot que se quiere vaciar. El jugador no puede utilizar el slot nueva.
 - Códigos de retorno:
 - 200 Si se ha eliminado la partida correctamente
 - 404 Si ha ocurrido un error (por ejemplo no existe partida guardada en el slot o se ha indicado un slot erróneo)
 - Contenido de retorno:
 - Ninguno
 - Llamada:
 - `http://puigpedros.salleurl.edu/pwi/pac4/partida.php?token={token}&slot={nueva,1,2}`

Requisitos: objetivos académicos

La práctica debe cumplir los siguientes requisitos:

- NO es necesario utilizar Bootstrap.
- Deben utilizarse CDNs (Content Delivery Network).
- Debéis entregar un .zip con:
 - Carpeta dev (con todos los archivos fuente y compilados).
 - Carpeta build (listo para jugar).
- El nombre del archivo .zip debe ser **GRUPO_NUMEROGRUPO_PAC4.zip**.

- Los documentos correspondientes deben validar tanto HTML5 como CSS con el validador de la w3c. No debe aparecer ningún error, aunque se acepta cualquier aviso warning o info.

Limitaciones: rompiendo reglas

Fuera de todas las anteriores especificaciones los desarrolladores sois libres de añadir todo aquello que consideréis necesario para hacer el juego más divertido, accionable y atractivo para los jugadores de este tipo de género.

Pueden añadirse nuevas funcionalidades como la raza (humano, enano, elfo...), sortilegios (bola de fuego, ceguera, huida, invocación...), clases de jugador (guerrero, hechicero, druida...), más niveles... La limitación está en vuestra imaginación. Incluso podéis modificar las imágenes predeterminadas.

Es imperativo revisar el apartado "Práctica 4 Tips" en el eStudy para más información, resolución de dudas y comentarios del profesor. En este apartado puede que se indiquen aspectos importantes a tener en cuenta para la buena resolución de la práctica.

Evaluación: rúbrica

Todo lo que no está especificado en esta rúbrica se define a discreción de los desarrolladores. Para que la práctica pueda evaluarse debe cumplir los siguientes puntos:

- Los archivos HTML, CSS, SCSS, JavaScript y Gulp propios deben contener comentarios claros. Si se cree imprescindible describir el desarrollo se hará al principio del main.js entre comentarios `/**/`.
- Los archivos HTML, SCSS, Gulp y JavaScript deben estar libre de errores.
- Los archivos HTML y CSS deben validar por W3 sin errores.

Si se cumplen los requerimientos anteriores se aplicará la siguiente rúbrica:

Categoría	3	2	1	0
		JUEGO		
Movimiento	Además añade nuevos.	Realiza todos los movimientos básicos.	Realiza alguno de los movimientos básicos.	No se mueve.
Objetos físicos	Además añade nuevas características a los objetos.	Usa los objetos guardados.	Guarda los objetos encontrados y se muestran correctamente.	No guarda los objetos encontrados.
Lucha	Además añade nuevas	No hace la lucha por turnos o no indica la	No empieza primero el jugador	No hace.

	funcionalidades como huir, esquivar...	vida de cada luchador.	o no indica quién está atacando	
Nivel / XP	Además añade nuevas funcionalidades.	Se muestra en el juego y se aplica en las características del jugador.	Se guarda en el JSON pero no muestra en el juego.	No lo contempla en el juego.
Jugabilidad	Además, añade distintos objetos y enemigos, con características diferentes los unos a los otros.	Juego con paredes, objetos y enemigos contra los que luchar. Que tenga cierta dificultad para ganar puntos de experiencia.	Juego con paredes y objetos . Fácil conseguir puntos de experiencia y subir de nivel.	Juego sin paredes, objetos ni enemigos.
Información de datos del jugador y estado de la partida.	Además muestra información adicional a la básica (porcentajes, número de enemigos muertos...)	Muestra la información actualizada a tiempo real.	Muestra la información o alguna pero no a tiempo real.	No muestra toda la información.
Diseño	Además añade animaciones y efectos (visuales y/o sonoros).	Cambia por completo la interfaz gráfica inicial.	Añade algún elemento visual a la interfaz básica.	Utiliza la interfaz gráfica inicial.
Nivel mazmorra	Además cambia de nivel a niveles más allá del contiguo.	Además añade más niveles.	Cambia de nivel al siguiente o anterior.	No cambia de nivel.
TECNOLOGIA				
API	Envía y recibe y hace tratamiento de los códigos devueltos.	Envía y recibe en JSON.	No envía o no recibe información.	No envía y no recibe información del servidor.
JSON	Hace uso de los datos tratados.	Hace tratamiento de los datos con JSON.parse	Hace tratamiento de los datos con eval().	No hace tratamiento de los datos.
SASS	El código SCSS contiene lógica de programación (condicionales, mixins, variables...)	El CSS compilado sí valida W3 y se acompaña con el código fuente SCSS.	El CSS compilado no valida W3 y se acompaña con el código fuente SCSS.	El CSS compilado no valida W3 o no hay CSS compilado.
JavaScript	Utiliza una librería JavaScript.	Utiliza todas las estructuras de	Utiliza algunas de las estructuras de	Errores consola.

		programación vistas en clase.	programación vistas en clase.	
HTML5	Valida W3 sin errores, aunque se aceptan warnings o infos.			No valida W3.
Gulp	Existe el archivo de configuración gulp y las tareas están bien configuradas.	Existe el archivo gulp de configuración pero está mal configurado.	Existe el archivo de gulp de configuración pero no contiene ninguna tarea.	No existe el archivo gulp de configuración.
Carpetas Src & Build	Entrega las dos carpetas necesarias y contienen los archivos correctos.	Entrega las dos carpetas necesarias pero no contienen los archivos correctos.	Entrega carpetas pero no existe la src o build.	No entrega carpetas.