Programación de Videojuegos en Lenguajes Interpretados

Grado en Desarrollo de Videojuegos

Examen Convocatoria Ordinaria Curso 24-25

17 de diciembre de 2024

► ***Lee detenidamente el enunciado completo antes de comenzar el examen.***

Índice

1.- Descripción de la tarea del examen

Se implementará un juego con una serie de mecánicas parecidas a una de las fases del juego arcade Balloon Fight, cuyo primer juego se publicó en 1986 (Figura 1).

Figura 1: Balloon Fight

En el examen implementaremos una versión un poco más simple donde habrá un menú y un nivel de juego (Figura 2). En el nivel tendremos al jugador con un único globo y varios enemigos que se moverán en dirección al jugador.

Tenéis un ejemplo interactivo del **resultado completo del examen** **aquí**.

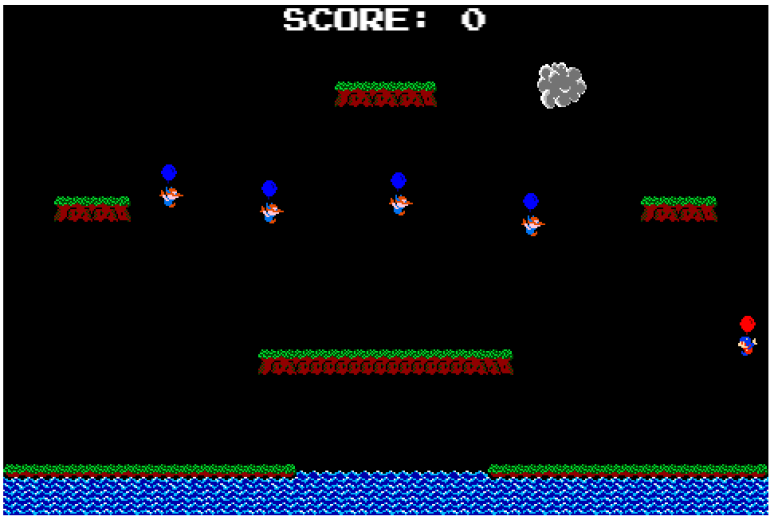

Figura 2: Nuestra versión de Balloon Fight.

Todo el material (sprites, sonidos, etc.) así como un esqueleto del HTML y el archivo game.js están disponibles en el Campus Virtual.

IMPORTANTE: Independientemente de la suma de puntos, el examen no estará aprobado si no hay una **aproximación funcional del juego donde el jugador pueda ganar o perder y se vuelva al menú**.

El exámen puede **aprobarse** siempre que el **código** sea **correcto y** haya una **aproximación** mínimamente **funcional** a las mecánicas fundamentales, es decir:

- Un menú de juego con un botón para poder jugar el nivel principal.
- Un jugador y enemigos que interactúen.
- Que el jugador pueda ganar o perder, según si su globo ha estallado o si todos los enemigos han sido vencidos.
- Una pantalla de victoria/derrota que regresa al menú.

Los apartados que aparecen a continuación no representan el orden de desarrollo del examen por lo que te recomendamos que los hagas de la manera que consideres más adecuada, teniendo en cuenta los mínimos que se piden.

Así mismo, te recomendamos que hagas uso de un repositorio de Git **local** en el que puedas ir **guardando versiones intermedias** de la solución del examen. El objetivo de esto es **que podáis deshacer cambios fácilmente** ya que es importante que **lo que entreguéis se ejecute**. Crear copias intermedias en local, aunque más rudimentario, también os puede servir.

1.1 Menú principal (0.5 puntos)

Cuando se empieza la partida, hay un menú con el título de juego en naranja (alineado a la derecha) y un botón "Start".

Si pulsamos con el ratón sobre el botón "Start", podremos empezar a jugar.

La fuente empleada para el título se llama "`balloonfont`" y la de los botones "`arcade`", ambas están ya cargadas desde el archivo CSS.

Figura 3. Menú de juego

1.2 Escena de juego (1.5 puntos)

Crea una escena con el fondo de juego. En la carpeta "`/assets`" tenéis los archivos "`terrain.json`" y "`tileset.png`" para poder generar la escena con tilemap.

El tilemap tiene dos capas "*land*" y "*water*" para plataformas y agua respectivamente. El *tileset* usado se llama "*tileset"*. Además en "`/assets/tilemapProject`" se encuentra el proyecto de Tiled.

Además del terreno del nivel, la escena de juego tendrá un contador de puntuación que se irá actualizando cuando se explotan globos y eliminan enemigos.

Figura 4. Escena de juego.

1.3 El jugador (1.5 punto)

El jugador está compuesto por dos elementos; el propio sprite del muñeco del jugador y el globo del jugador. El jugador puede moverse hacia los lados con las flechas del teclado (← →) y además volar con la tecla “Z”.

El jugador no puede moverse lateralmente si no está pulsando a la vez la tecla de volar. El jugador además tiene movimiento toroidal, al salir por un lado de la pantalla aparecerá por el otro. Sin embargo, no puede salir del nivel ni por arriba ni por abajo de este, colisionando con el borde superior e inferior.

El muñeco y su globo colisionan con las plataformas del escenario.

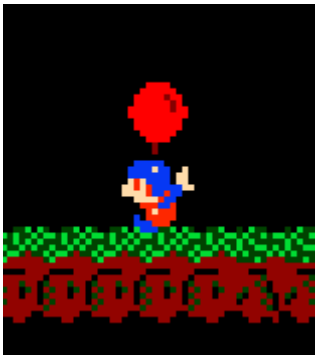

Figura 5. El jugador posicionado encima de una de las plataformas.

El jugador tiene dos spritesheets para el muñeco y uno para el globo.

Figura 6. El spritesheet del muñeco del jugador con las animaciones de movimiento y caída libre (player.png)

Figura 7. El spritesheet del muñeco con la animación de muerte (dead.png)

Figura 8. El globo del jugador y también de los enemigos (balloon.png)

El globo es pintado de rojo para diferenciarlos de los globos enemigos, que serán pintados de otro color.

1.4 Los enemigos (1.5 puntos)

Los enemigos, al igual que el jugador, están formados por dos sprite, el del muñeco y el del globo.

El nivel tiene varios enemigos que siguen al jugador, y cada uno de ellos lo hace a una velocidad aleatoria que se decide al iniciar el nivel. Los enemigos (tanto muñeco como globo) colisionan con las plataformas del nivel al igual que el jugador.

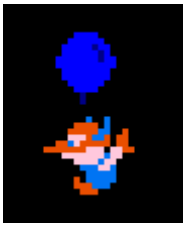

Figura 9. Enemigo.

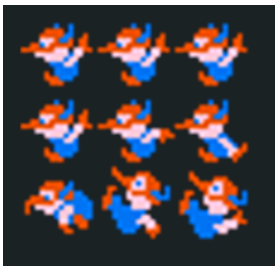

Figura 10. El spritesheet del muñeco del enemigo (enemy.png).

Cuando un enemigo pierde su globo, este caerá por la gravedad. Si el jugador golpea a un enemigo una vez ha perdido su globo, éste morirá. Por otro lado, si el enemigo no es golpeado durante los 10 segundos después de perder su globo, generará otro y volverá a volar siguiendo al jugador.

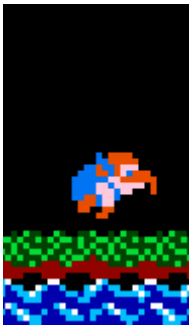

Figura 11. Enemigo cuyo globo ha estallado. El enemigo se queda quieto hasta que vuelva a ser golpeado por el jugador o bien pasen 10 segundos y genere un nuevo globo para seguir volando.

En el nivel habrá 4 enemigos dispersos por el mapa.

1.5 Interacción entre enemigos y jugador (2 puntos)

Si el muñeco del jugador golpea un globo de un enemigo, éste estallará, dejando al enemigo sin globo y cayendo por la gravedad. Después de un segundo, el enemigo volverá a ser vulnerable, y si es golpeado de nuevo mientras no tiene globo, morirá, desapareciendo del nivel.

Si el muñeco de un enemigo golpea el globo del jugador, éste estallará, dejando al jugador sin vida y perdiendo el nivel.

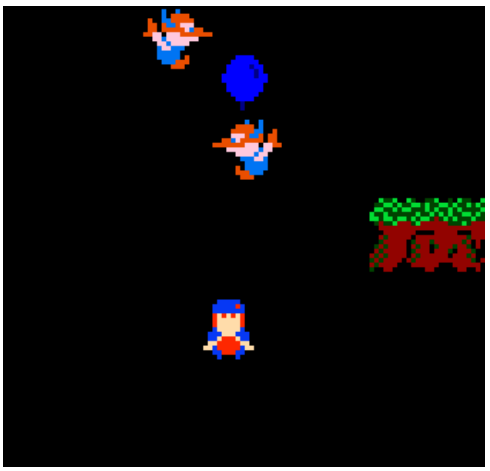

Figura 12. Jugador en caída libre y muerto después de que un enemigo haya explotado su globo.

1.6 Nube y rayo (1.2 puntos)

Para evitar que el jugador esté infinitamente consiguiendo puntos, después de 35 segundos de partida, una nube con una animación por código generará un rayo que puede matar al jugador si entra en contacto con él.

El rayo se mueve por todo el mapa y rebota en las plataformas.

Figura 13. Nube que genera el rayo (nube.png) y spritesheet del rayo en forma de estrella (star.png).

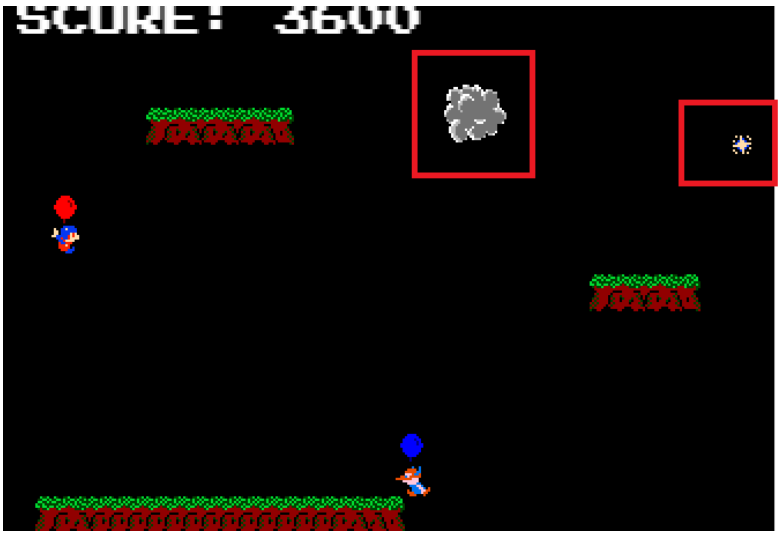

Figura 14. Escena con la nube y el rayo ya generado.

1.7 Puntuación (0.8 punto)

El jugador gana puntos por cada globo que explota de un enemigo (300 puntos) y por cada enemigo que elimina completamente del nivel (500 puntos).

La puntuación aparece en la parte superior de la pantalla actualizada en todo momento.

Figura 15. Partida con 1700 puntos conseguidos después de estallar 4 globos y eliminar un enemigo.

1.8 Victoria y derrota (1 punto)

Un partido del juego termina si se cumple una de las 2 condiciones siguientes:

- El jugador elimina a todos los enemigos por completo (el jugador gana la partida).
- El jugador pierde su globo (el jugador pierde la partida)

Cuando la partida termina se lanzará un sonido para indicar el final de partida, y después de 2 segundos, saldrá una pantalla con el mensaje de victoria y derrota. Además, en caso de ganar, se mostrará la puntuación conseguida.

Figura 16. Mensaje de Victoria/Derrota

Después de unos 3 segundos, se volverá a la escena inicial del título.

1.9 Sonidos y Animaciones (1 punto)

Los recursos que se proporcionan contienen 3 sonidos, 6 *spritesheets*, 2 imágenes, un *tilemap* y un *tileset* que se pueden utilizar para dar mayor vida al juego.

Sonidos:

- Sonido cuando explota un globo (balloon.mp3)
- Sonido cuando el jugador gana (victory.mp3)
- Sonido cuando el jugador pierde (loose.mp3)

La información para generar las animaciones se encuentran en el archivo **"*assets/sizes.txt*"**:

2.- Ayuda

Prioriza conseguir todas las mecánicas básicas (requisitos mínimos) para aprobar y mejora a partir de ahí.

Por ejemplo, un flujo de trabajo puede ser el de empezar por tener un menú simple con un botón que permita empezar el juego. Después hacer un **enemigo estático en la escena**. Lo siguiente sería tener al **jugador capaz de moverse**. Tanto el enemigo como el jugador tendrán dos sprites (un muñeco y un globo).

Después se puede abordar que ambos personajes (enemigo y jugador) interactúen, es decir, **que sean capaces de explotar el globo del contrario**. Una vez se tenga esto según qué globo ha estallado se puede **mostrar la escena de victoria o derrota** según quién haya quedado con el globo sobre su cabeza.

Lo siguiente pasos serían **abordar el escenario y la puntuación**, terminando con **mejorar la interacción entre enemigos y jugador** para cumplir con el resto de requisitos. Terminando con animaciones y sonidos.

Lo **último** que quedaría es añadir la **nube y generar el rayo**.

PD: El ejemplo del juego que se proporciona contiene varios trucos para facilitaros comprobar las mecánicas. Implementar **estos trucos no deben implementarse en el examen**, aunque puede ayudaros a vosotros mismos a comprobar el resultado conseguido. Las **teclas y trucos que podéis ejecutar** son los siguientes:

- "C" → Activa/Desactiva el modo debug para ver/ocultar los colliders.

- "P" → Destruye el globo del jugador.
- "O" → Destruye un enemigo.
- "I" → Generar rayo de la nube

3.- Evaluación

El código se ejecutará igual que se ha hecho durante el curso, abriendo un servidor web local en la raíz del proyecto (http-server o live-server, por ejemplo). Se probará con la última versión disponible de Google Chrome a fecha del examen, exclusivamente.

El examen tendrá una nota de 0 a 10 (aunque puede sacarse hasta 11 puntos), siendo necesario un 5 para aprobar. Si el código del examen no se ejecuta (error de sintaxis), el juego se queda colgado, o la solución es muy lejana a lo pedido, el examen estará suspenso.

Cada apartado recibirá, como máximo, el valor indicado. Y se valorará el estilo, el uso correcto de construcciones y orientación a objetos, y se tendrá en cuenta la solución en general (no sólo los apartados independientemente). Es decir, a partir de una versión que funcione y cumpla los mínimos pedidos, se tendrá en cuenta la calidad del código, tanto la arquitectura de clases como la corrección de la implementación, de las funciones y de los métodos.

4.- Entrega

La entrega se hará a través del Campus Virtual, en la entrega habilitada para tal propósito.

Se debe subir un proyecto completo como archivo comprimido llamado ***"balloon.zip"***. El proyecto deberá tener un archivo ***"README.txt"*** con el **nombre** del alumno y su ***DNI***. Un proyecto sin este archivo no será evaluado. La entrega es individual.

Una vez hecha la entrega (subida la solución al campus), descárgala y comprueba que contiene la última versión de tu examen (**asegúrate de que has subido la versión correcta**).

5.- Materiales

Todo el material para realizar el examen (esqueleto de HTML, `game.js`, sprites, sonidos…) está disponible en el Campus Virtual. Aunque este material es suficiente para hacer el examen, se puede modificar si se considera necesario.

Además el material contiene un archivo ***sizes.txt*** con el tamaño de las imágenes y de los frames de cada spritesheet.

Así mismo, se pueden usar todos los materiales disponibles en el campus y ejemplos y códigos propios (documentación, apuntes…).

No se puede establecer ningún tipo de comunicación con otros compañeros o personas externas. Tampoco está permitido el uso de IAs generativas. Cualquier de estas acciones se considerará copia y supone el suspenso en todas las convocatorias de la asignatura.

6.- Copia

Cualquier intento, fructuoso o infructuoso, de copia supondrá la aplicación de la normativa de la asignatura y el suspenso de la asignatura en todas las convocatorias restantes del curso actual.