Cuadernillo de prácticas
Motores de Videojuegos

Profesora:
Eva Ullán

Curso:
2020/2021

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Normas de entrega

Entrega de prácticas

- Cada práctica tendrá una fecha límite de entrega. No se admitirán prácticas entregadas fuera del plazo establecido.
- Las prácticas se entregarán exclusivamente a través del Campus Virtual, utilizando el mecanismo de entrega de prácticas.
- Únicamente un miembro del grupo de prácticas deberá encargarse de realizar la entrega. En el caso de que ambos componentes efectuasen el envío de la práctica, corregiremos aleatoriamente uno de ellos.
- Para realizar la entrega, será necesario crear un directorio (carpeta), llamado `GrupoNN`, siendo `NN` el número del grupo al que pertenece (con dos dígitos: 01, 02, ..., 40).
- En el directorio `GrupoNN` se incluirá un fichero `alumnos.txt`, en el que se indicará el nombre de los componentes del grupo que hayan intervenido en el desarrollo de la práctica.
- Además, se incluirán en la carpeta `GrupoNN` **únicamente** las carpetas `Assets`, `Packages` y `ProjectSettings` del proyecto de Unity.
- Se enviará un único archivo comprimido con formato .zip[1], llamado `GrupoNN.zip`, resultado de la compresión del directorio (carpeta) `GrupoNN`.
- No se tolerarán plagios[2] ni se permitirá hacer las prácticas entre varios grupos.

[1] Para comprimir el archivo puedes utilizar cualquiera de los compresores disponibles en el laboratorio con la condición de que generen ficheros zip. Ten en cuenta que el programa 7-zip puede generar ficheros .zip, pero para eso hay que indicar explícitamente que se desea ese formato.

[2] Los alumnos participantes en un caso de copia, de forma activa o pasiva, serán remitidos a la Comisión de Copias de la Facultad, la cual decidirá las sanciones a imponer.

Práctica 1: Lluvia de letras

Comienza por el comienzo —dijo, muy gravemente, el Rey— y sigue hasta que llegues al final; entonces paras.

Alicia en el País de las Maravillas
Lewis Carroll

Resumen: Nuestro primer juego en Unity
(Idea original de Marco Antonio Gómez Martín)

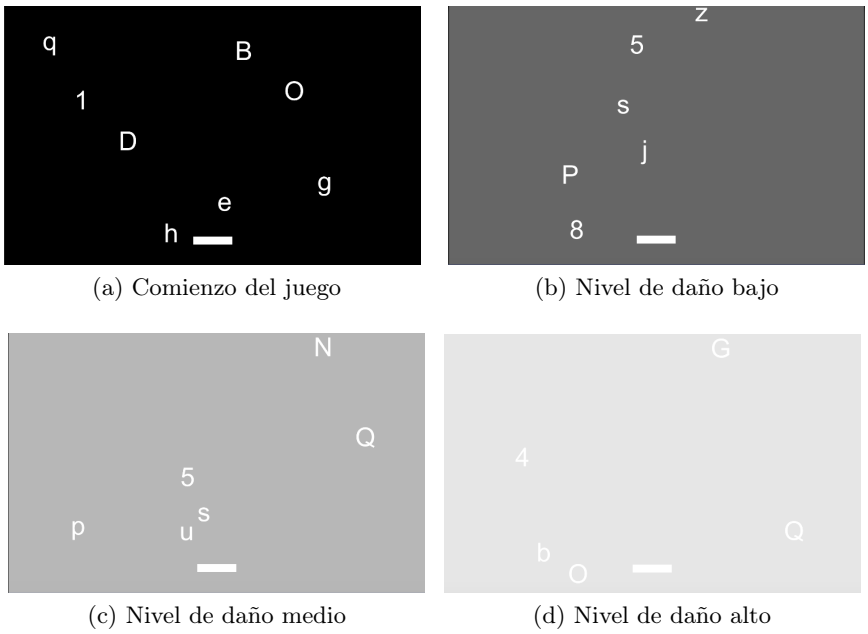
1.1. Descripción

Lluvia de letras es un juego colaborativo para dos jugadores, en el que van apareciendo letras en la parte superior de la pantalla, las cuales van cayendo en vertical hasta desaparecer, bien sea por llegar a la parte inferior de la pantalla o porque conseguimos "salvarlas".

Para rescatar a las letras existen dos mecanismos:

- El jugador con las manos en el teclado puede pulsar la tecla correspondiente a la letra. Si en la pantalla aparecen varias letras iguales, desaparecerán todas.
- El jugador con las manos en las flechas puede manejar la "pala" de la parte inferior para recoger las letras que el jugador que teclea no consigue eliminar.

Cuando una letra escapa a los jugadores y llega a la parte inferior, el nivel de daño aumenta. Cuando supera un umbral, la partida termina.



(a) Comienzo del juego

(b) Nivel de daño bajo

(c) Nivel de daño medio

(d) Nivel de daño alto

Figura 1.1: Distintos momentos del juego

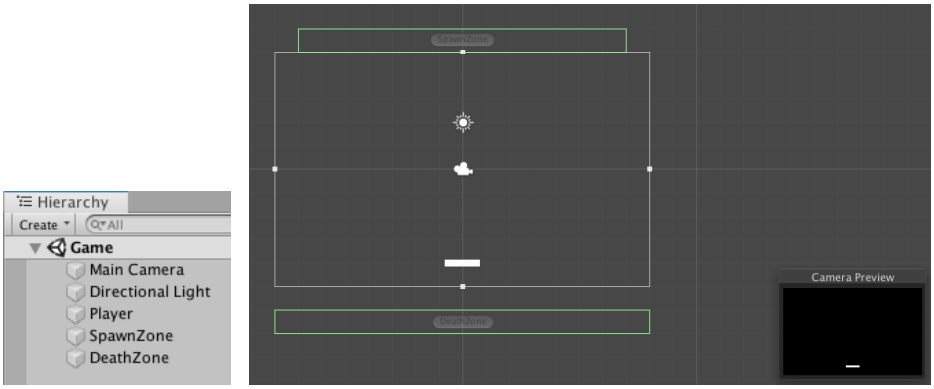
La ejecución comienza con un fondo negro. Las letras van apareciendo en la parte superior y bajando en línea recta a distintas velocidades.

El nivel de daño es nulo inicialmente, pero se va incrementando si los jugadores no son capaces de salvar a tiempo alguna de las letras. Cada vez que una letra llega al suelo (borde inferior de la pantalla), el nivel de daño se incrementa en un 10 %.

No es necesario incorporar ningún texto indicando el nivel de daño actual pues éste se verá reflejado con cambios en el fondo de la pantalla: el fondo comienza en negro y va acercándose al blanco puro a medida que el nivel de daño sube (como se puede ver en la figura 1.1).

Eso convierte al juego en difícil de superar con niveles de daño elevados, pues el fondo hace que las letras (blancas) sean difíciles de distinguir del fondo. Cuando el nivel de daño alcance el 100 %, el juego se volverá imposible de jugar y la partida terminará.

Para no hacer el juego imposible, la salud se repara con el tiempo, de forma que el nivel de daño desciende un 1 % por segundo. Esto hace que el propio fondo vaya acercándose de nuevo al negro si los jugadores consiguen aguantar un tiempo sin dejar caer por completo las letras.



(a) Vista Jerarquía (b) Vista Escena

Figura 1.2: Escena *Game*

1.2. Escena *Game*

El juego se desarrolla en una escena de Unity, la cual debe tener las siguientes entidades:

- **Main Camera**: la responsable del pintado. Está configurada en proyección *ortográfica*, con el fondo en negro y un tamaño de 5 unidades.
- **Directional Light**: la que pone Unity por defecto es suficiente.
- **Player**: una pala construida a partir de un cubo, con la que se pueden recoger las letras que caen.
- **DeathZone**: un cubo sin componente de dibujado, que establece la zona en la que se considera que las letras han bajado demasiado. Si esto ocurre, debe penalizarse a los jugadores aumentando el nivel de daño (y destruir la letra).
- **SpawnZone**: otro cubo sin componente de dibujado, que determina la zona en la que se crean las letras.

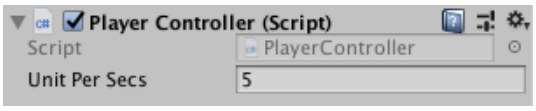
Destaca la ausencia en la escena de entidades para las letras. Éstas son, en realidad, creadas por el *Spawner* a partir de un *prefab*. El *prefab* **Letra** habrá sido creado previamente, a partir de un objeto dotado de un componente *Text Mesh*, que viene configurado de serie con un campo específico para contener texto. Para mejorar la calidad del texto, se debe configurar el componente con un tamaño de la fuente de 100 *(Font Size)* y un tamaño del carácter de 0.1 *(Character Size)*.

Por último, fijamos el *Aspect Ratio* a 16:10, de forma que nuestro mundo de juego tenga establecidos un ancho y un alto concretos.

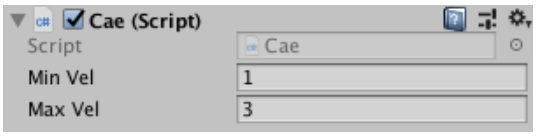
1.3. Implementación

Para crear el juego de forma *simple y rápida*, el proyecto no necesita muchos componentes. A continuación, se proporciona una descripción de los *scripts* a implementar y utilizar como componentes, junto con la indicación de las **únicas** variables configurables desde el Inspector. Todos los movimientos han de ser cinemáticos.

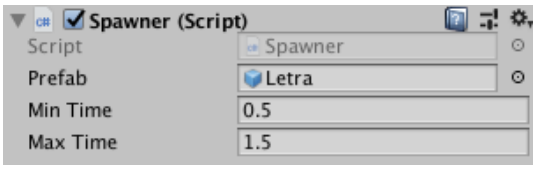
- `PlayerController`: Vigila las pulsaciones de teclado correspondientes al eje horizontal y mueve, en consecuencia, la entidad en el eje X, impidiendo que ésta sobrepase los bordes del mundo (teniendo en cuenta tanto el ancho del mundo como el de la entidad). Permite configurar la velocidad a la que se mueve.



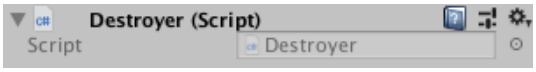
- `Cae`: Dota a la entidad asociada de la capacidad de descender (en el eje Y) a una velocidad constante aleatoria en un intervalo dado, configurable desde el editor. Si la velocidad es negativa, subirá en lugar de bajar.



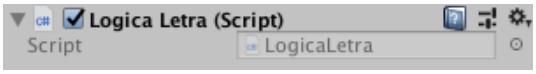
- `Spawner`: Permite crear copias de un objeto molde *(prefab)*, configurado desde el editor, en intervalos de tiempo aleatorios dentro de un rango dado. La posición X en la que se crea cada copia también es aleatoria dentro de los límites de la entidad a la cual esté asociado.



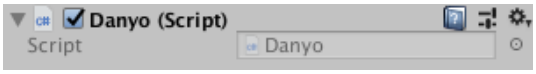
- `Destroyer`: Destruye a cualquier entidad que entre en la zona de influencia de la entidad a la cual esté asociado.



- `LogicaLetra`: En la creación del objeto al que esté asociado, asigna una letra de forma aleatoria al campo `text` del componente de tipo `Text Mesh`. Además, vigila las pulsaciones del teclado para autodestruirse cuando se pulse la tecla correspondiente.



- `Danyo`: Gestiona el nivel de daño, incrementándolo en 10 por cada letra perdida, partiendo de un daño inicial nulo. Paralelamente, la salud se recuperará a razón de 1 por segundo, vigilando que el nivel de daño nunca sea negativo. Modifica el fondo de la pantalla haciendo que el color de fondo de la cámara vaya acercándose al blanco cuanto mayor sea el nivel de daño. Cuando el nivel de daño supere el 100 %, la partida debe finalizar.



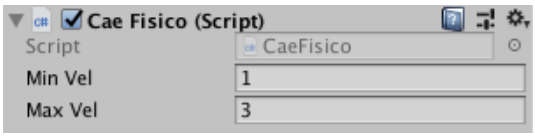
1.4. ¿Y si...?

Con lo que ya tenemos hecho, y lo que hemos ido aprendiendo mientras tanto, no podremos evitar caer en la tentación de replantearnos la implementación del juego, preguntándonos qué pasaría si las letras fueran objetos físicos. Para no quedarnos con la duda, vamos a ponernos a ello, evitando estropear el juego que ya tenemos funcionando.

Empezaremos creando una nueva escena *PhysGame*, duplicando nuestra escena *Game* plenamente funcional. Cualquier cosa que hagamos en esta escena no afectará a la anterior (siempre que no cambiemos la implementación de los *scripts* que ya funcionan).

Crearemos un nuevo *prefab* **LetraFisica**, duplicando el anterior, y configurándolo como un objeto físico al que no le afecte la gravedad. Le asociaremos un nuevo componente `CaeFisico`, en sustitución de `Cae`.

- `CaeFisico`: Dota a una entidad física de una velocidad aleatoria constante que la hace descender con respecto al eje Y, dentro de un intervalo dado, configurable desde el editor.



Aunque `CaeFisico` tiene casi la misma especificación que `Cae`, la diferencia fundamental a la hora de implementarlo estriba en que lo vamos a usar para mover a un objeto físico, y no a uno cinemático. Por lo demás, estamos interesados en que su descenso se consiga dotándole de una velocidad fijada aleatoriamente en el momento de su creación.

La idea de este apartado es conseguir una variante física del juego sin alterar su filosofía: ambos jugadores colaboran en el rescate de las letras, mientras que las letras no salvadas incrementan el nivel de daño. Debemos procurar conseguir un funcionamiento lo más parecido posible al ya conseguido en el apartado anterior.

1.5. Opcional

Dado que en las escenas anteriores no ha habido espacio para la creatividad, llega el momento para poder darle rienda suelta, proponiendo en una nueva escena *MyGame* la incorporación de **algún detalle diferencial** que cambie el funcionamiento del juego (sin estropear el de las escenas anteriores).

No se trata de reimplementar nada, ni de hacer un juego nuevo, sino de aprovechar la inmersión para dar un paso más, que nos evite parar en seco la divagación acerca de qué pasaría si... hiciéramos algún otro cambio que influyera en la experiencia de juego.

Procuraremos controlarnos para no emplear demasiado tiempo en este apartado, e incluiremos en la entrega un archivo `descripcion.txt` en el que explicaremos lo hecho y cómo ha cambiado el comportamiento del juego.

1.6. Entrega

La entrega se realizará respetando las indicaciones realizadas al principio del presente documento.

Fecha límite: Sábado 28 de noviembre a las 16:00 horas