

Futuristic Balls

Creación de un juego con HTML5, Multimedia, URJC.

Marco Caballero del Dedo.
Pedro Espinosa Ferrández.

m.caballerod@alumnos.urjc.es
p.espinosa@alumnos.urjc.es

Tabla de contenidos

I.	Introducción	3
	Pantallas	4
	Bienvenida.....	4
	Carga	5
	Selección de nivel.....	6
	Canvas por nivel	7
	Final.....	7
	Elementos/personajes.....	8
	Ground	8
	Blocks	9
	Villains	9
	Heroes.....	10
	Conclusiones.....	11
	Autores	11

Introducción

Este proyecto trata de cumplir los requisitos propuestos en la primera práctica de la asignatura Multimedia del grado en Ingeniería de Software (URJC, 4º curso), por lo tanto incluye lo necesario para construir un juego con HTML5 basado en jQuery y Box2d.

Para ello se ofrece un juego que emula la mecánica del famoso *Angry Birds* (Rovio Entertainment) pero con un nuevo entorno y personajes, en este caso, unas bolas futuristas que tratan de aniquilar los nuevos artilugios de comida basura del futuro. En este juego llamado **“Futuristic Balls”**, y que consta de cuatro niveles con dificultad en aumento, podemos diferenciar los distintos elementos clasificándolos de la siguiente manera:

- Pantallas
 - Bienvenida/Inicio
 - Carga
 - Selección de nivel
 - Canvas por nivel
 - Final.
- Elementos/personajes
 - Ground
 - Blocks
 - Villains
 - Heroes
- Estados
 - Intro
 - Load-next-hero
 - Wait-for-firing
 - Firing
 - Fired

Pantallas

Bienvenida

Se muestra al comienzo del juego, contiene el logo con el título del juego y permite mediante un botón (*play button*) iniciar el juego y transitar a la pantalla en la que se muestra la selección de nivel. El diseño es el siguiente:

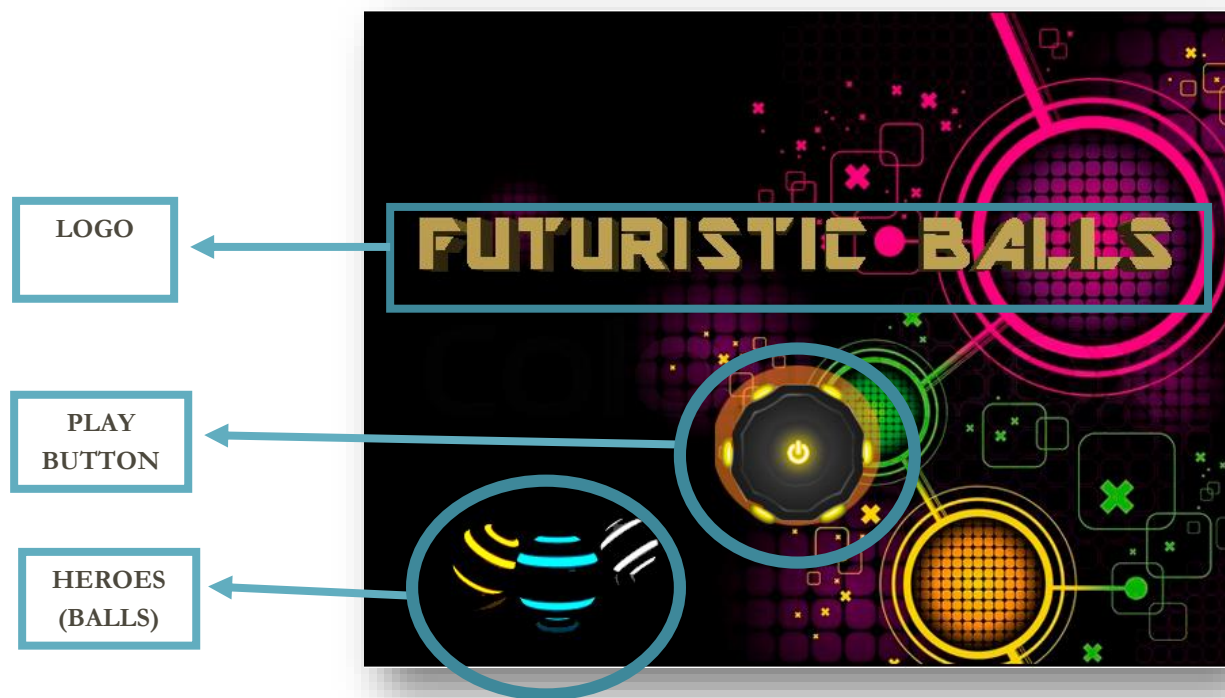


Ilustración 1- Pantalla inicial

Logo

Futuristic Balls 3D, creado por nosotros con Photoshop (Adobe Photoshop): 2015.5.1.

Play button

Template open-source de <https://sylnvapht.deviantart.com/art/Futuristic-Organic-Button-175357782> modificado con Photoshop (Adobe Photoshop): 2015.5.1.

Heroes (balls)

Template de www.lightingstores.eu modificado con *Photoshop* (Adobe Photoshop): 2015.5.1.

Carga

Se muestra en las transiciones entre pantallas, contiene un “loader spinner”, un gif que muestra el nivel de carga con una animación, que incluye el porcentaje de carga sobre el fondo de juego inicial. El diseño es el siguiente:



Ilustración 2 - Loader en pantalla principal

Loader

Asset *loader.gif* provisto en los assets del AulaVirtual (<https://www.aulavirtual.urjc.es>), disponibles para la práctica y guión de creación de un juego con html5.

Selección de nivel

Permite seleccionar el nivel al que se quiere jugar, pudiendo elegir entre 4 diferentes cada uno con mayor dificultad. El diseño es el siguiente:



Ilustración 3 - Pantalla de selección de nivel

Levels

Button de selección de nivel, creado con *Photoshop (Adobe Photoshop): 2015.5.1* utilizando de nuevo los héroes (*balls*) para darle más customización al botón.



Ilustración 4 - Botón de selección de nivel

Canvas por nivel

Los diferentes niveles se sirven de las estructuras disponibles para ofrecer diferentes escenarios y dificultades al jugador. En el apartado elementos/personajes se detallarán más a fondo. A modo de ejemplo, el diseño del nivel 3 es el siguiente:

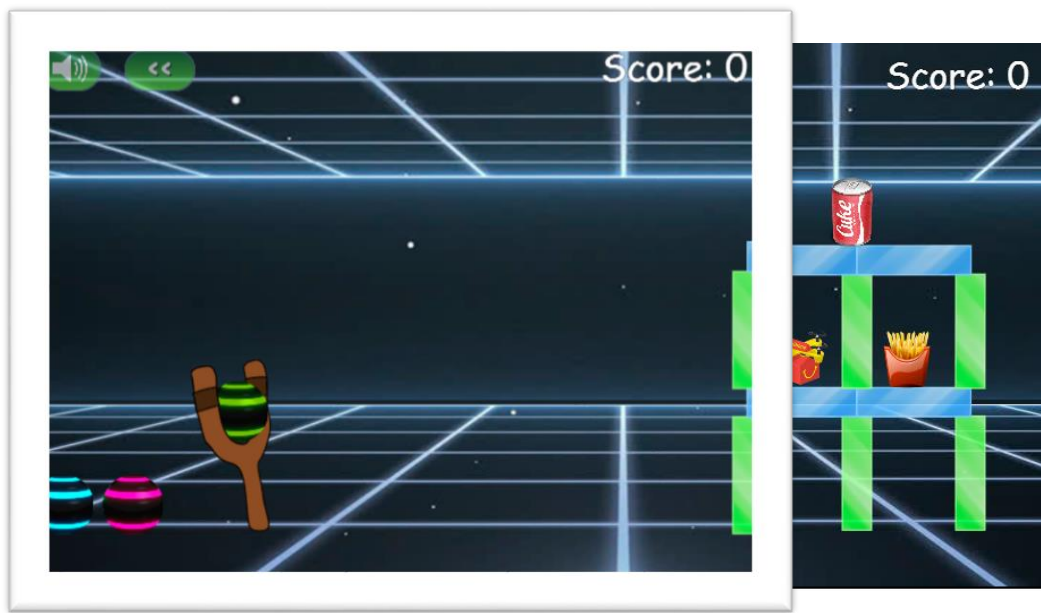


Ilustración 5 - Ejemplo de nivel (nivel 3)

Final

Pantalla de final, muestra el resultado del nivel y permite avanzar al siguiente, repetir el nivel o volver al menú de selección de nivel. Hay una pequeña variación en función de si es el último nivel o no, pero el diseño es el siguiente:



Ilustración 6 - Final de nivel

Elementos/personajes

Ground

Suelo, objeto estático que mantiene los elementos sin que caigan al vacío, además del tirachinas que sostiene las bolas a lanzar. El diseño es el siguiente:

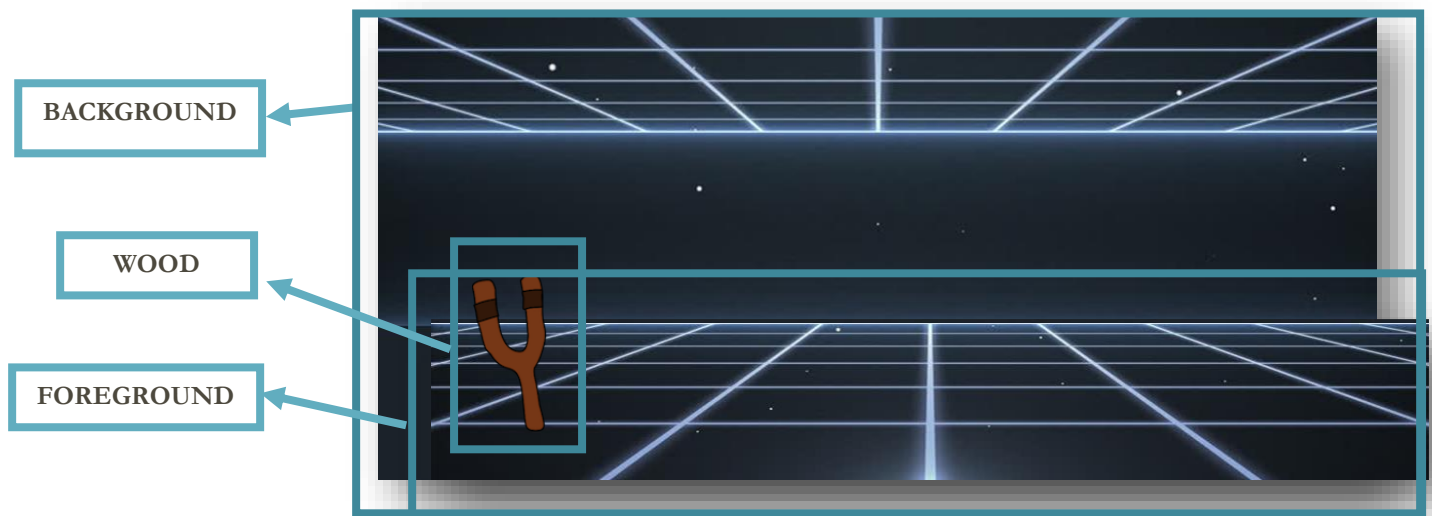


Ilustración 7 - Elementos básicos del nivel

Background:

Imagen de fondo sobre la cual se muestra otra que simula el suelo y la parte que da sensación de relieve (*foreground*) gracias al contraste entre ambas, y el tirachinas (llamado “*wood*” en el juego) que permite lanzar las diferentes bolas que se ofrecen en cada nivel.

Foregound

Se muestra sobre el background y simula el suelo, además se ofrece un efecto parrallax para dar sensación de profundidad

Wood

Se muestra en el comienzo del foreground, y sostiene la bola que se lanzará en el turno actual.

Blocks

Objetos con los que formar una estructura que sostenga los villanos (villains), además de romperse al caer ofrecen un sonido que simula la rotura de un cristal. Se han usado dos tipos de bloques, cristal azul y verde futurista:



Ilustración 9 - Bloque verde




Ilustración 8 - Bloque azul


Villains

Objetos a destruir, nos otorgan puntos en base a las calorías que tienen, se han reutilizado algunos villanos ya existentes y ofrecidos en el Aula Virtual y se ha creado uno para que la temática futurista sea adecuada. Los diferentes villanos y sus características son las siguientes:

Sodacan

Slogan	Calorías	Nombre	Diseño
Bebida azucarada peligrosa para la salud, y que ha de ser destruida	150	sodacan	

Fries

Slogan	Calorías	Nombre	Diseño
Patatas con mucha mucha grasa, destrucción inmediata necesaria.	350	fries	

McDrone

Slogan	Calorías	Nombre	Diseño
Nuestro mayor enemigo, un dron que transporta comida basura.	500	McDrone (futuristic_burger)	

Heroes

Son los personajes con los que jugará el jugador, se lanzan contra las estructuras para destruir a los enemigos.

Balls

El prototipo de héroe elegido para esta temática es una bola futurista destructora, se ofrece en 3 colores modificado con Photoshop (Adobe Photoshop): 2015.5.1. Los diseños son los siguientes:

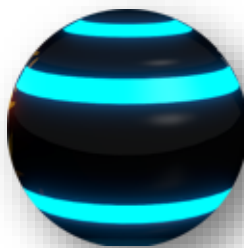


Ilustración 12 - Blue ball



Ilustración 11 - Pink ball

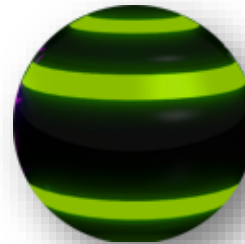
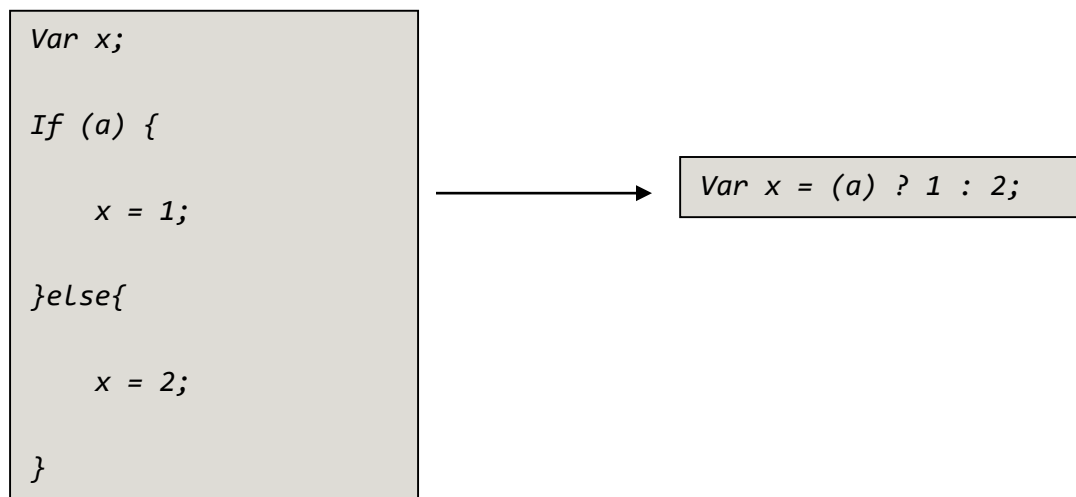


Ilustración 10 - Yellow ball

Conclusiones

Como se propuso, hemos aprendido a utilizar el objeto canvas de *HTML5* así como los diferentes objetos que deberían componer un buen juego actual, además se ha aprendido a utilizar la librería Box2d de JavaScript en colaboración con *jQuery*. Por otra parte, es cierto que tener todo el código en el mismo fichero, sin una orientación a objetos ni arquitectura adecuada haría de este proyecto un juego 'inmatenible' a largo plazo, y por lo tanto, sería necesaria una gran refactorización para eliminar smell codes y antipatrones así como para otorgar un patrón a toda la aplicación. En este caso *MVVM* (model view – view model) parece el más adecuado, pero un *MVC* o *MVP* podrían ser válidos.

Además, se han modificado ciertas expresiones, condiciones anidadas y otros patrones de código a las actuales expresiones lambda, objects literals y otras mejoras que ofrece JavaScript desde *ecmascript6*. A modo de ejemplo:



Autores

- *Marco Caballero Del Dedo*,
 - o m.caballerod@alumnos.urjc.es
 - o <https://www.linkedin.com/in/marco-caballero/>
- *Pedro Espinosa Ferrandez*,
 - o p.espinosa@alumnos.urjc.es
 - o <https://www.linkedin.com/in/pedro-espinosa-fer/>

