Práctica 3

Diseño e implementación de un diseño gráfico

Jonathan Martín Valera

Marcos Avilés Luque

1. Descripción

Nuestro pequeño sistema gráfico consiste en una aplicación web simulando un juego implementado con Three.js.

Este juego consiste en un tablero, el cual posee objetos en su interior. Mediante el uso de los cursores del teclado podremos inclinar el tablero y mover la pelota por fuerza de gravedad.

El objetivo principal es colisionar con el objetivo con una pelota. Este objetivo estará rodeado de obstáculos que irán restando vida, y otros que permitirán recuperarla.

Mientras la vida no llegue a 0 se podrá seguir jugando e intentar pasar de nivel.

Al colisionar con el objetivo, se aumentará el nivel, creando una nueva pelota adicional en el tablero y nuevos obstáculos, dificultando así la posibilidad de ir incrementando de nivel cada vez más y más.

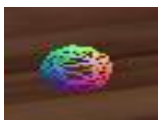
Los objetos que están dentro del tablero son:

- **Pelota:** Objeto que se moverá por la fuerza de la gravedad a medida de que inclinamos el tablero. Este objeto es el más importante, ya que tendremos que centrarnos en éste para que colisione con el objetivo y pasar de nivel.

- **Cono rojo:** Este objeto restará salud del jugador cuando alguna de las pelotas colisione con él.

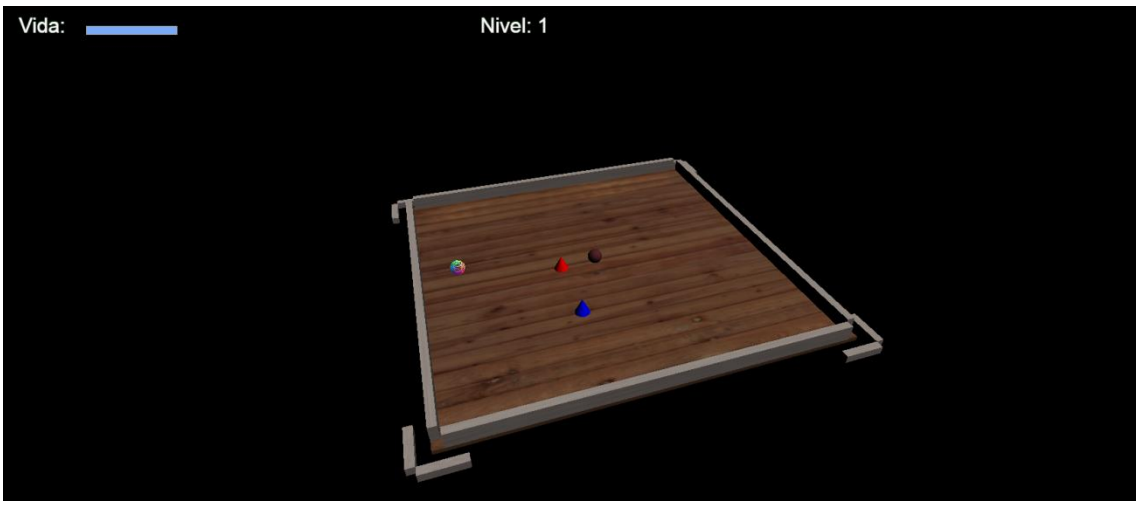
- **Cono azul:** Este objeto restaurará salud del jugador cuando alguna de las pelotas colisione con él.

- **Objetivo:** Objeto con el que la pelota tiene que colisionar para ir subiendo de nivel.

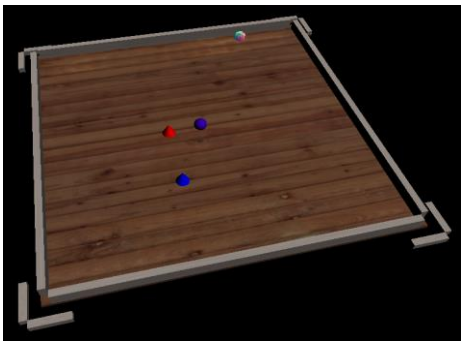


La posición de los objetos es determinada de forma aleatoria dentro del tablero, por lo que cada nivel es diferente en cada ejecución. La dificultad radica en el número de obstáculos y pelotas que se crean, siendo cada vez más probable la colisión de cualquiera de ellas con cualquier obstáculo penalizador.

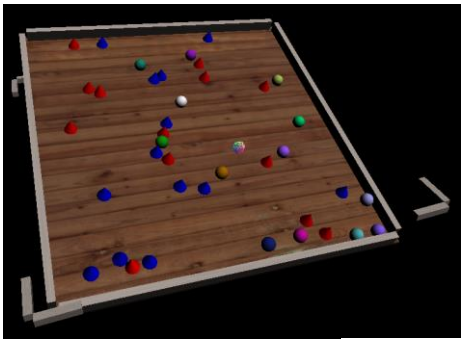
La interfaz que se nos presenta al ejecutar el juego es la siguiente:



Como se puede comprobar, tenemos una barra que indica la salud, un texto que nos indica el nivel en el que estamos, y el tablero con el que va a interactuar el usuario.

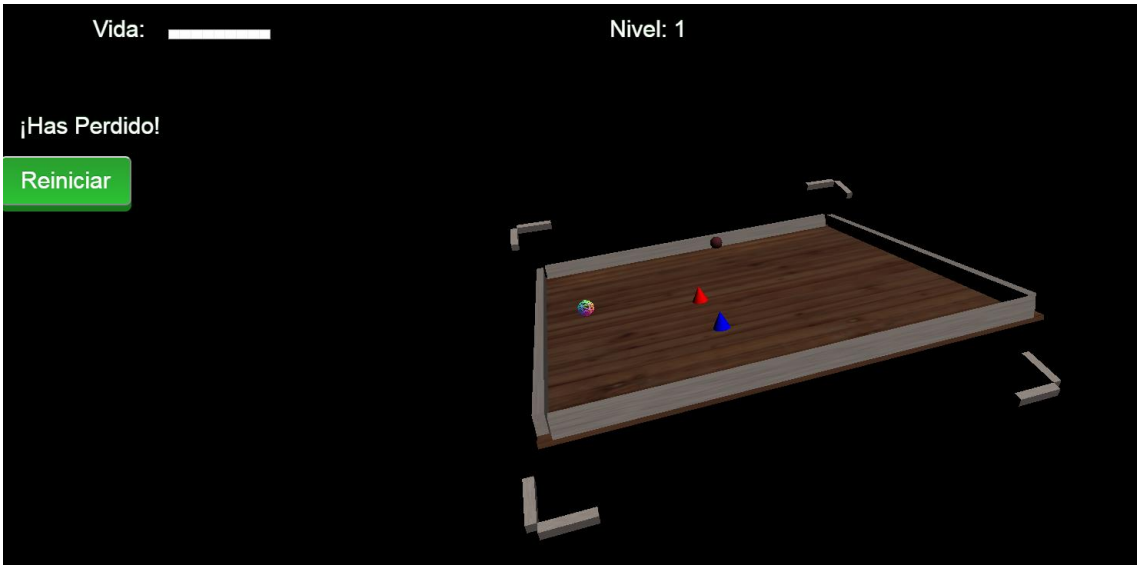


Nivel 1



Nivel 13

Cuando agotamos la salud, se nos presenta la siguiente interfaz:



Al pulsar en el botón de reiniciar, automáticamente empezaremos desde el nivel 1 con la salud al máximo.

El objetivo del juego es alcanzar el máximo nivel posible, superando cada vez el record que haya hasta ese momento.

2. ¿Cómo se ha desarrollado?

Para el desarrollo de esta práctica, hemos utilizado la librería: **Physijs.**

Este complemento se puede añadir fácilmente a three.js, ya que se siguen todas las convenciones normales de ésta. Physijs se construye sobre ammo.js y ejecuta la simulación física en un hilo separado para evitar disminuir el rendimiento de la aplicación y aumentar el tiempo de renderizado 3D.

Las características principales son:

- Soporte para múltiples formas de objetos.
- El material tiene un control sobre la fricción y restitución.
- Detección de colisiones y eventos integrados.
- Sistema de vehículos.
- Sistema de restricción completo incluyendo puntos, grados de libertad…
- Rotaciones utilizando sistemas de Euler o quaternion.
- Construido sobre three.js para mantener la convención y estilo de codificación.

Esta aplicación la hemos desarrollado en varios módulos:

Pincho.js: Contiene la función que nos permite crear un objeto que restará o restaurará salud en una posición determinada.

Pelota.js: Contiene la función que nos permite crear un objeto pelota en una posición determinada. El color de ésta será elegido de forma aleatoria.

Obstaculo.js: Contiene la función para crear las paredes que delimitarán el tablero y nos permitirán guiarnos en la inclinación de éste. Nos permite crearlo con una posición, tamaño, peso y textura determinada.

Objetivo.js: Contiene la función para crear el objeto objetivo con una posición determinada.

Main.js: Programa principal que realiza la construcción de la escena, junto con todas las llamadas y funciones para realizar el renderizado de la imagen, cámara, luces, objetos y la interacción con el usuario.

En este módulo hemos definido unas funciones para controlar la creación de objetos y el desarrollo de la partida. Éstas son:

- **iniciarPartida() :** Crea el tablero, las paredes del tablero, un techo (objeto dummy), los objetos que suman y restan vida, el objetivo y la pelota.
- **borrarObjetos():** Elimina de la escena el conjunto de objetos creados en iniciarPartida().
- **cambiarEstadoColision():** Función usada para el control de la colisión. Modifica una variable de control que se utiliza para evitar que no se tengan en cuenta las múltiples colisiones en un tiempo determinado.
- **comprobarObstaculo(objeto):** Esta función es llamada al producirse una colisión entre la pelota y cualquier objeto. Comprueba si el objeto con el que se ha colisionado es bueno, malo o es el objetivo; en cada caso tiene una salida distinta.
- **soldar(obstaculo, sujeccion):** Nos permite añadir una restricción física entre 2 objetos para que no se separen.
- **setLife(value):** Actualiza en la interfaz de usuario la vida del jugador. En caso de que sea 0, presenta el botón de reiniciar la partida.
-

3. Referencias

1. Documentación y ejemplos sobre Physijs: http://chandlerprall.github.io/Physijs/
2. Github con librerías usadas: https://github.com/chandlerprall/Physijs