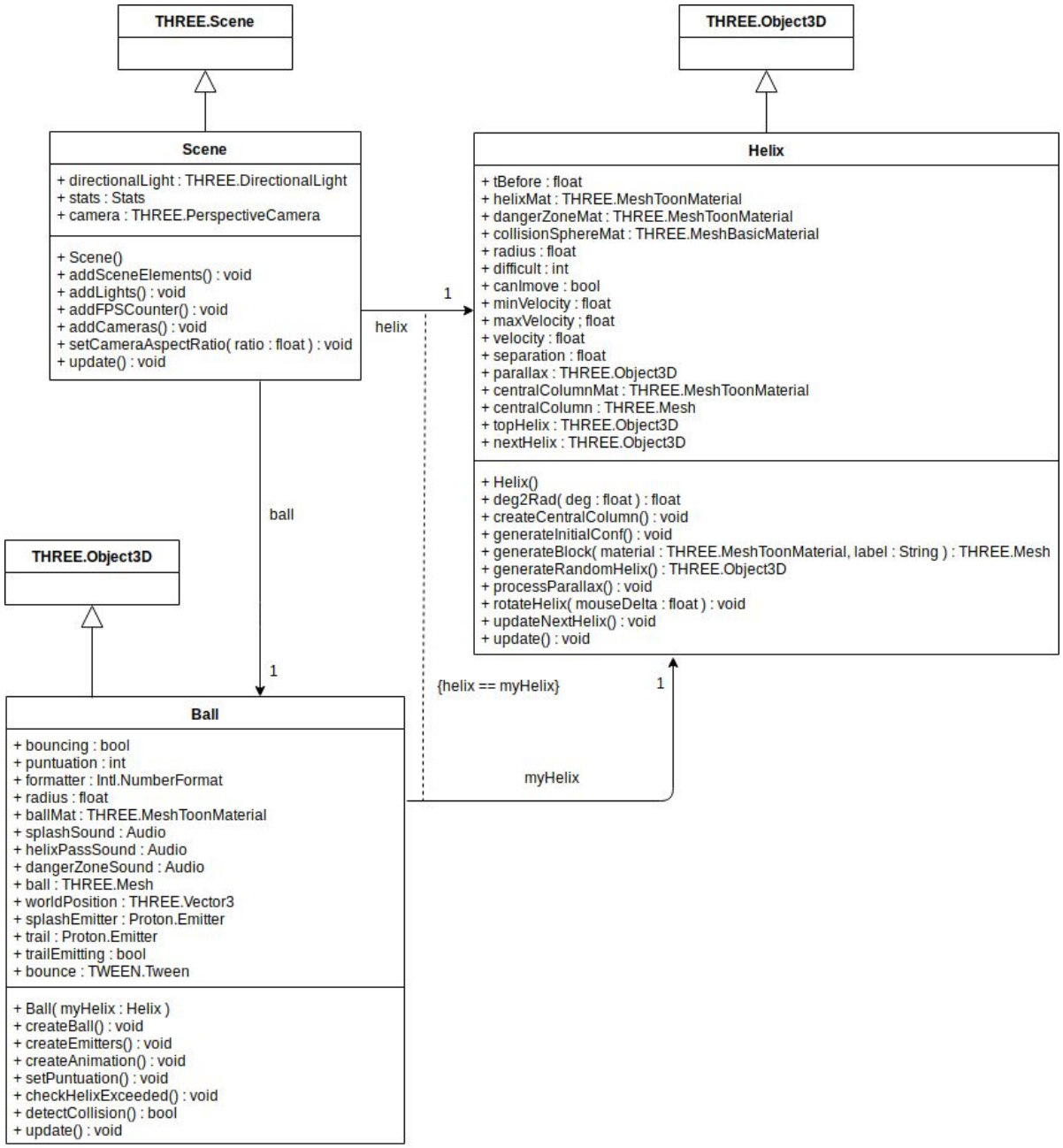
Sistemas Gráficos

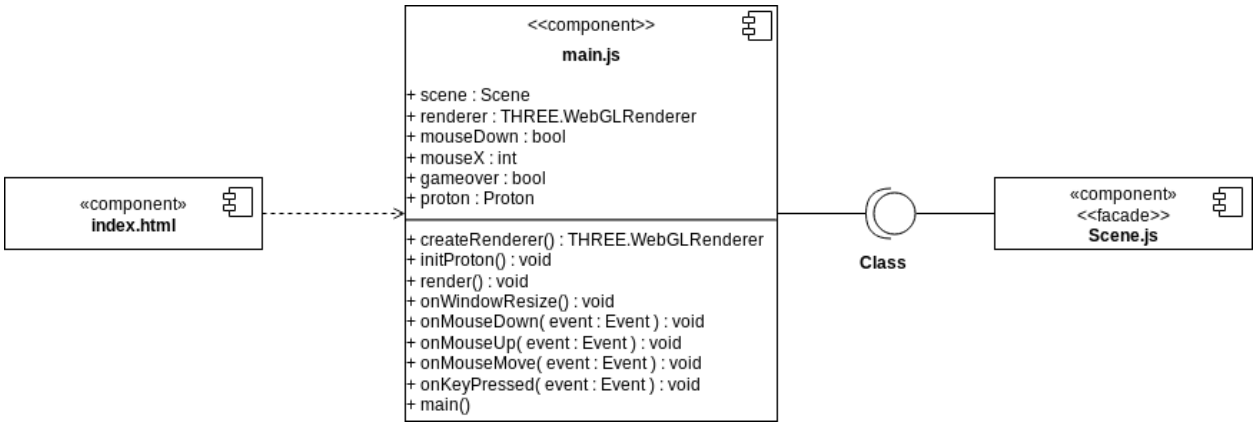
Álvaro Fernández García

Documento de Diseño

Helix Jump

DIAGRAMA DE CLASES





DESCRIPCIÓN

Este apartado incluye una breve descripción de aquellos aspectos más relevantes de las clases y scripts implementados:

- **Clase Helix.js:** Como su propio nombre indica, esta clase incluye todo lo relacionado con el procesamiento de la hélice. Una de las decisiones que se tomó durante el diseño fue que, en lugar de hacer que la bola caiga a través de un escenario infinito, está permanecerá estática en el centro de la pantalla y serán todas las hélices las que se desplacen hacia arriba (generando un parallax infinito), dando la sensación de que la pelota se precipita. Esto facilita tanto la destrucción de aquellas partes del escenario que ya han sido superadas (optimizando la memoria y realizando el mínimo número de cálculos posibles) como la creación de nuevos obstáculos. Solo es necesario comprobar si la hélice más alta ha sobrepasado un límite. Si esto es cierto, se elimina y se añade una nueva por la cola. Tampoco tenemos que preocuparnos por mover la cámara, esta siempre permanece fija enfocando a la bola.

  La torre de hélices no es la única que se genera de forma procedural, sino también las propias hélices individuales. Están constituidas por un conjunto de bloques (semicilindros de 45º) dispuestos de forma aleatoria en 8 posibles posiciones (45º * 8 = 360º). Durante la generación se respetan unas restricciones: debe haber entre 1 y 4 huecos y nunca más de dos consecutivos.

  Para determinar el número de zonas de peligro (aquellas que te hacen perder si las tocas) se utiliza un entero que representa la dificultad. Dicho número nos sirve para calcular la probabilidad de que un bloque sea o no de peligro. La dificultad aumenta conforme el jugador consigue puntuación.

La clase también realiza una aproximación de la aceleración provocada por la gravedad (aumentando la velocidad del parallax progresivamente mientras la pelota no colisione) y es la encargada de rotarse a sí misma según la variación horizontal del ratón.

- **Clase Ball.js:** esta clase es mucho más sencilla que la anterior. Crea la bola y la coloca centrada en la pantalla, inmóvil.

  Tiene una misión fundamental: detectar si se ha producido o no una colisión con la hélice. Cuando se produce una colisión pueden suceder dos cosas: si ha colisionado contra una zona de peligro, se debe parar la simulación y notificar a la clase escena de que se ha producido el fin de juego. Si era una zona segura, debe reproducir la animación de rebote (hecha con Tween), junto a las partículas (hechas con la biblioteca Proton, que se explica en el otro documento) y los sonidos correspondientes. Para saber de qué tipo era el bloque, se utiliza el campo “name” del Object3D.

  Si no se ha producido colisión, quiere decir que ha superado la hélice, por lo que debe actualizar la puntuación del jugador, la dificultad si procede y continuar cayendo.

  Para la detección de colisiones se ha implementado un sistema propio. En el centro de cada bloque se coloca una esfera y se realiza el test de intersección esfera-esfera contra la bola. Para optimizar estas comprobaciones, se mantiene una referencia a la próxima hélice con la que se encontrará la pelota y solo se realiza el test frente a ella.

- **Clase Scene.js:** Hereda directamente de la clase escena de Three. Crea la cámara, las luces, añade el contador de FPS e incorpora las dos clases anteriores a la escena.

  Su método update llama a los métodos update de la clase Ball y Helix y muestra la información de fin de juego cuando recibe el mensaje correspondiente por parte de la pelota.

- **Script main.js:** Se trata del main de la aplicación.

  Contiene variables globales, crea el render, inicializa el sistema de partículas, enlaza los event listener del teclado y ratón y realiza el renderizado de la escena.

- **index.html:** Además de incluir las bibliotecas y los ficheros de Javascript, define los campos de texto para la puntuación y GameOver, crea la fuente que se utiliza en los mismos y establece el color de fondo para la aplicación mediante un gradiente de dos colores.

GRAFO DE ESCENA

A continuación se muestra el grafo de escena resumido de la aplicación: