Práctica Final – Breakout en Three.js

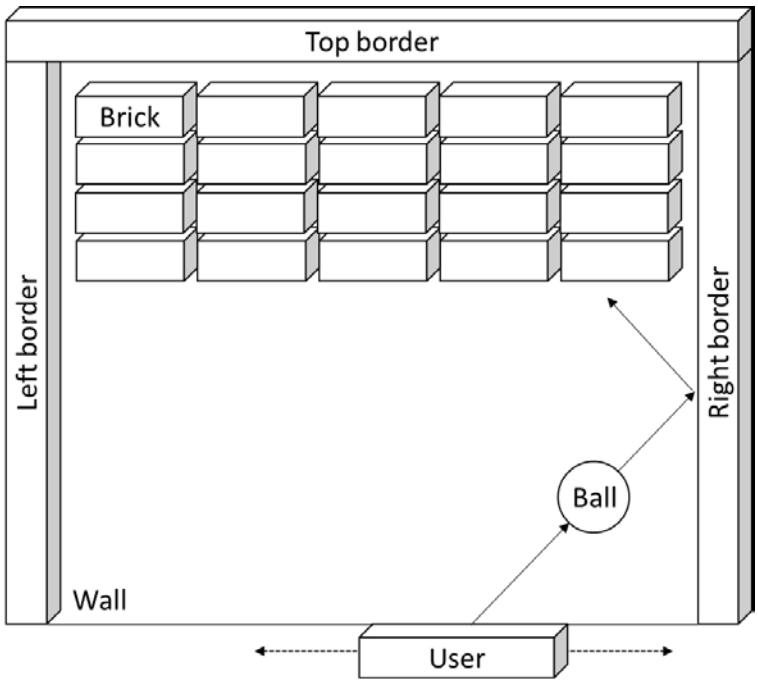
El objetivo de esta práctica es desarrollar una aplicación web que contenga un gráfico 3D interactivo creado con Three.js. Este gráfico implementará el juego clásico: **Breakout**.

Puntos totales posibles del ejercicio: 10

Funcionamiento básico

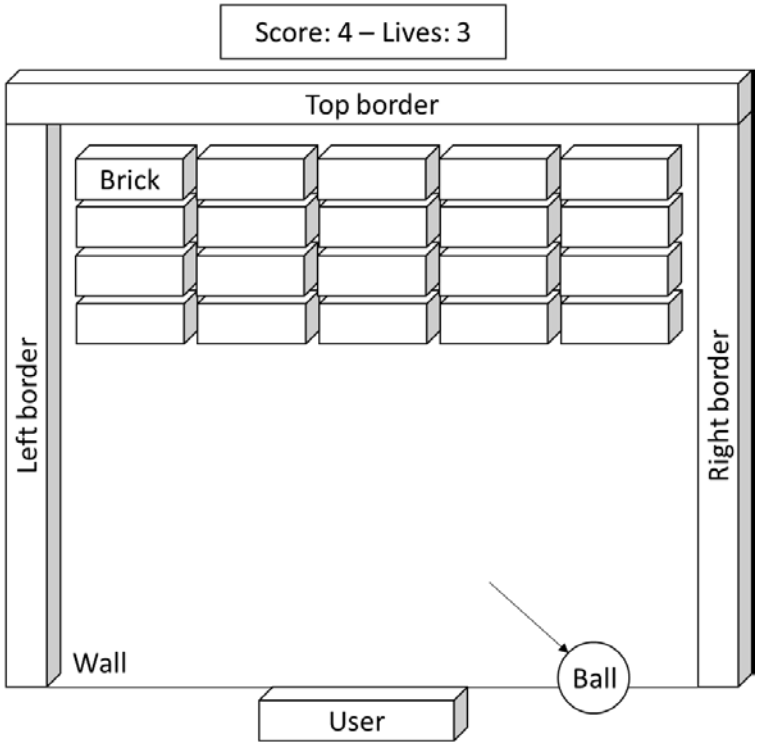
El gráfico 3D que implemente el juego estará formado por los siguientes elementos:

1. Pared. Superficie plana en 2D por la que se desplazará la bola.
2. Bola. Esfera en 3D que se utilizará en el desarrollo del juego.
3. Borde izquierdo, derecho y superior. Hexaedros en los laterales del plano de juego (pared).
4. Ladrillos: Hexaedros situados en la zona de juego (5-7 columnas y 4-5 filas).
5. Usuario. Hexaedro situado en la parte inferior del plano. Su movimiento en el eje X será realizado por el jugador, capturando las pulsaciones de teclado de las flechas izquierda y derecha.



La bola se desplazará por la pared variando la posición de sus coordenadas X e Y. Existirá detección de colisión de la bola con respecto a los bordes derecho, izquierdo y superior, así como con respecto a los hexaedros de usuario y los ladrillos. El juego consistirá en devolver la bola por parte del usuario, desplazando el respectivo hexaedro, hasta que todos los ladrillos hayan sido eliminados. Si el hexaedro de usuario bloquea esta posición, la bola rebotará en la dirección contraria. En caso contrario, la bola saldrá del plano y se restará una vida al jugador.

Por ejemplo:



En algún lugar de la página web deberá existir un marcador que muestre la puntuación y el número de vidas. En el funcionamiento básico, no es necesario que sea dentro del gráfico Three.js (se puede hacer con elementos HTML de tipo texto). Se empezará con 0 puntos y 3 vidas, y se sumará un punto por cada bloque eliminado y se restará una vida por cada bola perdida. Si el jugador se queda sin vidas o elimina todos los bloques, el juego finalizará.

Al cargar la página el juego no empezará hasta que el usuario pulse una tecla (por ejemplo, la barra espaciadora). En ese momento la bola empezará a moverse hacia arriba y en una de las 2 direcciones del plano (izquierda o derecha). El hexaedro del usuario se moverá a la izquierda o derecha en función de la pulsación de las teclas ← y → del teclado. Siempre que el usuario elimine un bloque o pierda una vida se actualizará el marcador, y en caso de perder una vida, además el juego se parará hasta que el usuario pulse de nuevo la tecla de inicio. Al perder las 3 vidas o al eliminar todos los bloques, se declarará ganador (si ha eliminado todos los ladrillos) o perdedor (si pierde todas sus vidas) y se ofrecerá la opción de volver a jugar.

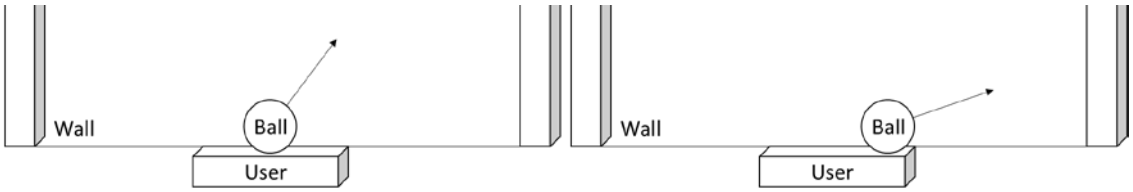
El modo de funcionamiento básico supone que se usarán los siguientes elementos:

- Texturas para todos los objectos (pared, bordes, ladrillos, usuario y bola).
- Fuente de luz direccional o puntual. Además, La bola deberá producir efectos de sombra, que será reflejada tanto en la pared, como en los bordes y hexaedros.

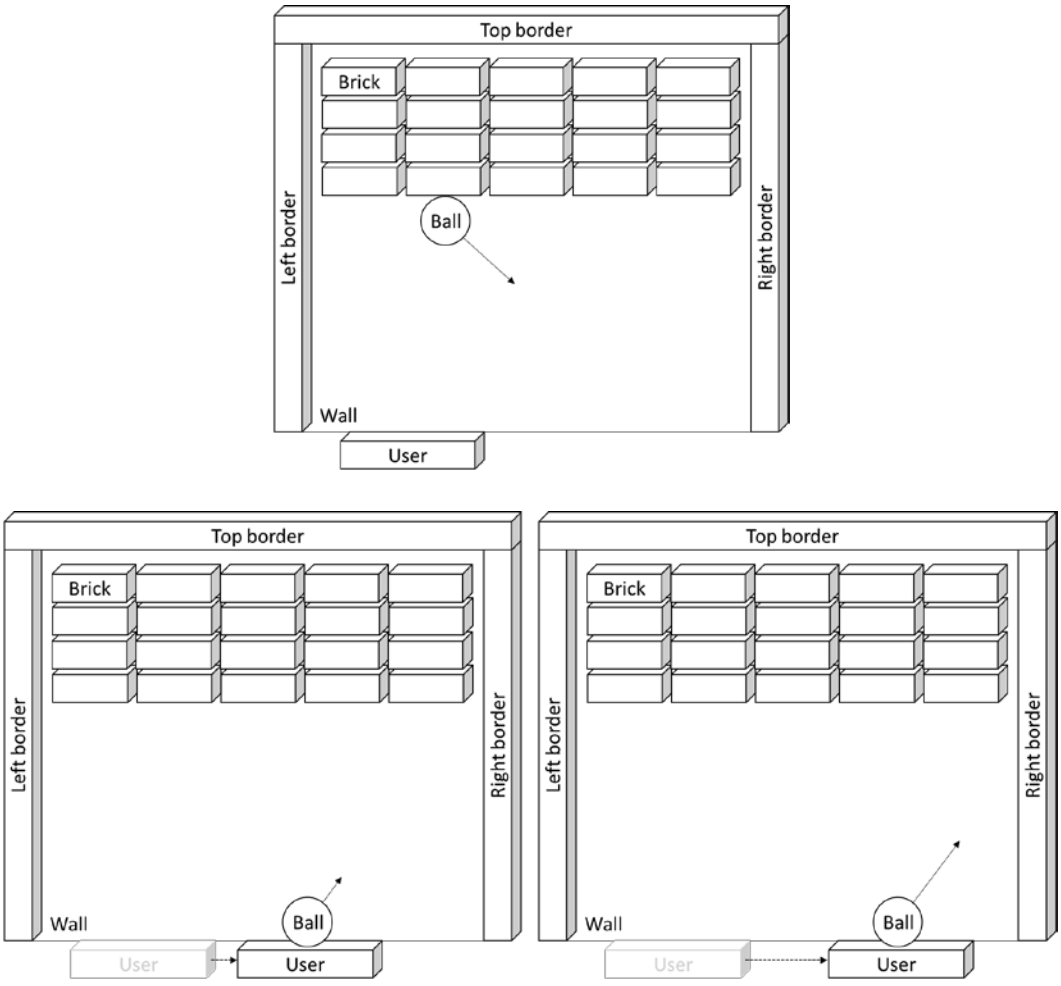
Funcionamiento avanzado

En el funcionamiento básico se supone que la velocidad con la que la bola se desplaza verticalmente (esto es, como se mueve a lo largo del eje Y) y el ángulo con el que rebota en los bordes, ladrillos y jugadores (esto es, como se mueve a lo largo eje X) es constante. En el funcionamiento avanzado estos 2 parámetros serán variables.

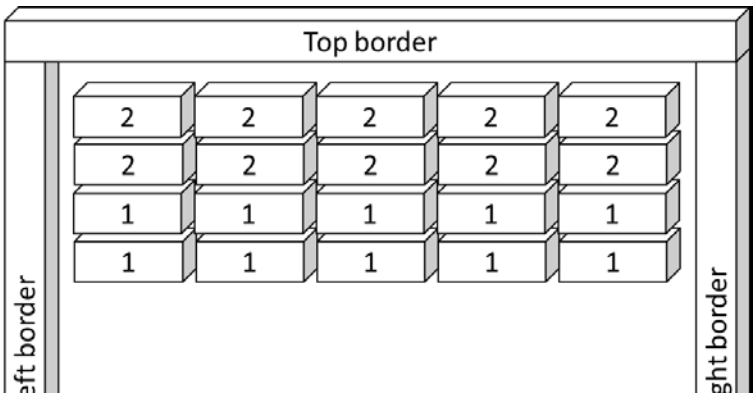
Para variar el **ángulo de rebote**, se tendrá en cuenta la posición del hexaedro con respecto a la bola a la hora de bloquearla. El ángulo será mayor si se bloquea con el extremo del hexaedro, y menor si se bloquea de forma centrada.



Para la **velocidad de desplazamiento vertical** se tendrá en cuenta el desplazamiento del hexaedro de su posición inicial (cuando rebota la bola sobre un ladrillo) respecto al punto donde se bloquea por el usuario. A mayor desplazamiento horizontal, mayor velocidad de desplazamiento vertical.

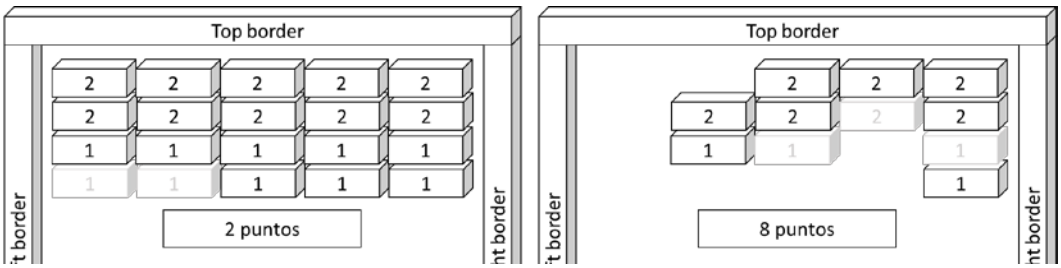


Además, para que sea un juego más interesante, se tendrá en cuenta el número de bloques eliminado en cada jugada y su posición. De este modo, cada bloque que inicialmente tiene un valor de puntuación de 1, tendrá un valor distinto dependiendo de su posición vertical. La fila más cercana al usuario, tendrá un valor de 1, el cuál se verá incrementado en 1 unidad por cada dos filas (a más filas, más puntuación).



Además, si el número de bloques eliminados entre un rebote de la bola con el usuario y el siguiente es mayor que 3, la puntuación obtenida en esa jugada valdrá el doble.

Por ejemplo:



Mejoras

Se deja abierto una parte de calificación (ver sección evaluación) para introducir mejoras extras en la práctica. Las posibilidades de mejora pueden ser variadas. Se calificarán las mejoras que mejoren la experiencia de uso de la aplicación. Algunas ideas son:

- Introducir niveles de dificultad. Pueden ser configurables al iniciar el juego o ir aumentando según se juega una y otra vez.
- Marcador con gráfico 3D en lugar de como texto HTML.
- Posibilidad de elegir diferentes texturas para los elementos de juego (“skin”).
- Animaciones adicionales (movimiento de la cámara o punto de luz en función del juego).
- …

Para optar a la nota máxima (10 puntos) habrá que implementar al menos dos mejoras extras.

En caso de duda sobre las mejoras que serán aceptadas, antes de implementar nada, siempre consultar con el profesor para tener el visto bueno.

Evaluación

Esta práctica se evaluará con un máximo de 10 puntos. La distribución de estos puntos será de la siguiente manera:

- Funcionamiento básico: 5 puntos.
- Funcionamiento avanzado: 3 puntos.
- Mejoras: 2 puntos.

Además, se tendrá en cuenta la calidad de código. Se supone que el código deberá cumplir los siguientes principios:

1. Se deben evitar los fragmentos de código duplicados (principio DRY, Don't Repeat Yourself)
2. El código debe ser legible. Esto se traduce en:
    a. Usar nombres de variables, funciones, etc. significativos (preferiblemente en inglés).
    b. El código debe estar correctamente indentado (usando tabs o espacios).
    c. No se utilizará código innecesario (por ejemplo, funciones que no se usan, condiciones que no se cumplen, etc).

Si se detectan problemas de calidad de código en base a estos dos principios, se restará hasta 1 punto de la nota obtenida al sumar las componentes anteriores (funcionamiento básico, avanzado, y mejoras).

Consejos

Es aconsejable seguir las siguientes recomendaciones:

- Comenzar implementando la funcionalidad básica. Una vez realizada, si se aspira a sacar más de 5 puntos, se puede implementar la funcionalidad avanzada, y por último las mejoras extras.
- Ir paso a paso. La mejor estrategia sería implementar pequeños cambios comprobando y verificando en cada momento que todo funciona como se espera.
- Usar la consola de depuración del navegador (Chrome, Firefox) para visualizar las trazas en la consola JavaScript (console.log) así como los posibles errores según vamos desarrollando.

Entrega

La entrega se realizará de forma **individual**. Hay que subir un archivo comprimido .zip en el aula virtual con todos los ficheros de la práctica, esto es:

- Página web (fuente HTML y JavaScript).
- Imagen(es) de textura.
- Fuentes (estilo CSS, texto JSON, sonido MP3, etc.)

La práctica será evaluada en primer lugar ejecutada desde un navegador Chrome y servida con un servidor web local.

En la cabecera de la página web deberá aparecer un comentario (esto es, un párrafo entre los símbolos <!-- y -->) informando de las partes que has implementado y tu nombre. Por ejemplo:

```
<!--
Autor: José Miguel

Partes implementadas:

- Funcionalidad básica
- Funcionalidad avanzada:
      - Ángulo de rebote variable
      - Velocidad de desplazamiento vertical variable
- Mejoras:
      - Niveles de dificultad

 -->
<!DOCTYPE html>
<html>

<head>
<title>Breakout</title>
```

El ejemplo anterior aspiraría a sacar una máxima de 8 puntos (5 de la funcionalidad básica, 2 de la funcionalidad avanzada y 1 de las mejoras).