LABERINTO BABYLON.JS

REALIZADO POR:

ISABEL MARÍA JIMÉNEZ CUMBRERAS

MÓNICA GARCÍA ROSA

ASIGNATURA: REALIDAD VIRTUAL

PROFESORES: JOSE CARPIO CAÑADA y FRANCISCO JOSE MORENO VELO

INTRODUCCIÓN

La práctica realizada consiste en la realización de un juego que tiene como protagonista un laberinto que debe ser cruzado. En cada partida el jugador será colocado en la entrada del mismo y debe atravesarlo en el menor tiempo posible llegando a la salida. Veremos al comenzar el juego un cronómetro que indicara en cada momento el tiempo transcurrido finalizando su actividad en el momento en que se llegue a la salida.

El laberinto está compuesto por muros que en su mayoría poseen la misma textura, solo algunos de ellos están forrados con carteles que permitirán ubicar al jugador en un punto concreto en cada momento. Además, en algunos lugares encontraremos cubos o esferas como objetos también de orientación para el jugador.

El juego dispone de una vista aérea gracias a la que es posible observar desde un punto superior todo el laberinto y conocer concretamente la ubicación del jugador. Ayudará a la persona que juegue en ese momento a conocer su posición y la salida.

Además, se dispone de un botón de instrucciones y una zona de descarga de mapas (estructuras de laberintos).

En cuanto a los mapas que se pueden adjuntar, se añade en la zona de descarga una plantilla a partir de la que el propio jugador puede generar sus mapas personalizados. Esta plantilla incluye instrucciones pertinentes para la realización de los laberintos.

PUNTOS CONSEGUIDOS EN LA PRÁCTICA

De los puntos que se habían propuesto para realizar en la práctica estos son los que se han conseguido:

ELEMENTOS BÁSICOS	CONSEGUIDOS
Crear un modelo 3D navegable con los cursores a partir de una definición del laberinto en un fichero de texto, una imagen, etc	SI
El sistema debe detectar las colisiones con los muros	SI
Debe incluir con el código del proyecto un manual que explique cómo se crea el laberinto a partir de un fichero de texto, imagen, etc	SI
ELEMENTOS ADICIONALES	CONSEGIDOS
Alguna forma de saber si el laberinto se ha resuelto[1]	SI
Medir tiempos de resolución del laberinto[2]	SI
Permitir la inclusión de objetos de referencia de forma sencilla (planta, mueble, etc)[3]	SI
Permitir alguna forma de quitar muros y dejas solo los elementos de referencia	NO
Permitir el uso de un casco de realidad virtual tipo Oculus para visualizar el modelo utilizando alguna de las librerías disponibles	NO
Integrar los sensores del casco de realidad virtual con la navegación en el laberinto	NO
Permitir la inclusión de fondos (montañas, edificios, etc.) que puedan servir como elementos de referencia extra laberínticos	NO
Permitir conocer el camino que ha seguido un usuario en la resolución del laberinto[4]	SI
Crear un juego a partir de la aplicación que permita hacer un ranking de tiempos de resolución de los laberintos.	NO

[1] Se incluye identificación de salida y es posible observar la posición del jugador.

[2] Se incluye cronómetro y se detecta el momento en el que se llega a la salida mostrándose el tiempo empleado en la realización del recorrido.

[3] Se incluyen esferas y cubos para referenciar posiciones.

[4] Se usa la posición del jugador.

FICHEROS Y CARPETAS DEL JUEGO

La práctica está compuesta por usa serie de carpetas y archivos que se explican a continuación:

Carpetas:

- **Css**: Almacena el fichero de estilos CSS3 (estilos.css) que permiten que los botones se muestren con unas características y posicionamiento concretos.
- **Instrucciones**: Almacena el fichero de instrucciones en formato PDF que podrá ser accesible desde el botón **Instrucciones del juego**.
- **Js**: Almacena el fichero javascript que genera el juego. Contiene elementos propios de BABYLON.JS y estructuras propias de javascript. Explicaremos sus funciones a continuación.
- **Mapas**: Contiene dos mapas, laberintos, de ejemplo. Además un fichero en formato docx que se puede usar como plantilla en la generación de nuevos laberintos. En el fichero además se dan indicaciones de cómo deben realizarse estas plantillas. Veremos este mismo archivo en formato PDF. Para finalizar la carpeta incluye el fichero RAR con los archivos mencionados para su descarga desde la página web.
- **Texturas**: Contiene los archivos de imagen PNG o JPG que se utilizan como texturas de paredes y suelos.

Ficheros:

- **Index.html**: Fichero principal que hace llamada al archivo javascript y css para la generación de la interfaz.
- **Estilos.css**: Fichero de estilos CSS 3.
- **Laberinto.js**: Fichero javascript que incluye las junciones BABYLON.JS y Javascript para la generación del juego y su entorno.

FICHERO INDEX.HTML

El contenido de este fichero se muestra a continuación:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Laberinto Mónica García Rosa e Isabel María Jiménez
Cumbreras</title>
    <meta charset="utf-8">

    <script src="js/qrcode.js"></script>
    <script src="js/hand.minified-1.2.js"></script>
    <script src="js/cannon.js"></script>
    <script src="js/oimo.js"></script>
    <script src="js/babylon.js"></script>

    <link rel="stylesheet" type="text/css" href="css/estilos.css"
media="screen" />

</head>
<body>
    <div name="cambioMapa"><input type="file" id="cargaMapa"
name="cargador"/></div>
    <div name="IniciarLaberinto"><input type="button" id="inicio"
name="iniciar" value="Comenzar Juego"/></div>
    <div name="cambioVista"><input type="button" id="vistaSuperior"
name="vistaSup" value="Vista superior/Vista normal"></div>
    <div name="ayuda"><input type="button" id="botonAyuda"
value="Instrucciones del juego"/></div>
    <div name="reloj" id="reloj">00:00:00</div>
    <div name="autores" id="autoras">
        <h2>Autoras: Mónica García Rosa - Isabel María Jiménez
Cumbreras</h2>
    </div>
    <div name="descargaMapas" id="mapas">
        Descarga aquí mapas de ejemplo y la plantilla para realizar
mapas personalizados
        <a href="mapas/mapas.rar" target="_blank"> [Mapas +
plantilla] </a>
    </div>
    <canvas id="canvas"></canvas>
    <script src="js/laberinto.js"></script>
</body>
</html>
```

A destacar de este fichero:

```
<script src="js/qrcode.js"></script>
<script src="js/hand.minified-1.2.js"></script>
<script src="js/cannon.js"></script>
<script src="js/oimo.js"></script>
<script src="js/babylon.js"></script>
```

En esta zona se realizan las llamadas a los diferentes archivos del Framework BABYLON.JS que serán necesarios para usar determinadas funciones de generación del laberinto.

```
<link rel="stylesheet" type="text/css" href="css/estilos.css"
media="screen" />
```

Enlace al fichero de estilos CSS3 estilos.css

```
<div name="cambioMapa"><input type="file" id="cargaMapa"
name="cargador"/></div>
    <div name="IniciarLaberinto"><input type="button" id="inicio"
name="iniciar" value="Comenzar Juego"/></div>
    <div name="cambioVista"><input type="button" id="vistaSuperior"
name="vistaSup" value="Vista superior/Vista normal"></div>
    <div name="ayuda"><input type="button" id="botonAyuda"
value="Instrucciones del juego"/></div>
    <div name="reloj" id="reloj">00:00:00</div>
    <div name="autores" id="autoras">
        <h2>Autoras: Mónica García Rosa - Isabel María Jiménez
Cumbreras</h2>
    </div>
    <div name="descargaMapas" id="mapas">
        Descarga aquí mapas de ejemplo y la plantilla para realizar
mapas personalizados
        <a href="mapas/mapas.rar" target="_blank"> [Mapas +
plantilla] </a>
    </div>
```

Conjunto de botones de la interfaz. Se usan capas (etiqueta <DIV>) para la colocación de estos. Por orden estos son los diferentes botones:

1. Cargar mapa.
2. Comenzar juego.
3. Cambio de vista aérea o normal.
4. Ayuda, instrucciones del juego.

Además se incluyen otros elementos como:

1. Cronómetro.
2. Relación de autoras de la práctica.
3. Zona de descarga de mapas y plantilla.

Así, el resultado de codificar esta zona será en la web el que se muestra en la siguiente imagen.

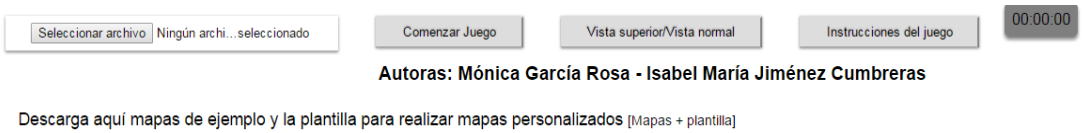

Figura 1. Resultado del cógido HTML estudiado.

```
<canvas id="canvas"></canvas>
<script src="js/laberinto.js"></script>
```

Generación del canvas que no es más que el objeto HTML5 que albergará el entorno gráfico. Justo después se hace referencia al fichero javascript que se ha creado y formará todo el juego. Este enlace puede incluirse igualmente en la cabecera del documento, entre las etiquetas <HEAD> y </HEAD>

FICHERO LABERINTO.JS

Este fichero contiene todas las funciones que generan el juego. Antes de nada vamos a explicar la función a partir de la que se obtiene la información del mapa. Las estructuras de los laberintos se encuentran almacenadas en ficheros de texto TXT. Estos ficheros tienen siempre la misma composición. Poseen un total de 40 filas y 40 números por fila, es decir sus dimensiones son 40x40. Cada línea contiene sus números separados por comas. Los valores para estos números pueden ser 0, 1, 2 o 3. También podemos encontrar una letra E o una letra S.

Significado de los valores del fichero de texto:

- 0 → Representa un espacio vacío en el laberinto.
- 1 → Representa un muro.
- 2 → Representa un objeto caja que se coloca en el laberinto para orientar al jugador.
- 3 → Representa un objeto esfera. Su finalidad es similar al objeto caja.
- E → Representa la entrada al laberinto.
- S → Representa la salida del laberinto.

Así, un fichero mapa puede contener algo similar a lo que se muestra a continuación:

```
1,0,0,E,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1
1,0,0,1,1,1,1,1,0,1,0,1,1,1,0,1,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1
1,1,0,0,1,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,3,0,0,0,1,0,1,0,0,0,0,1,0,0,0,0,1
1,0,0,0,1,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,0,1,0,0,1,1,1,1,1,1,0,1
1,0,0,1,1,2,0,0,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1
1,0,0,1,0,0,0,0,1,0,1,0,1,0,1,1,1,1,0,2,1,0,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,0,1
1,0,0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1
1,0,0,0,0,0,1,1,1,1,1,0,0,1,0,0,0,1,0,1,1,0,1,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1
1,0,1,1,1,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,1
1,0,1,1,1,0,2,0,0,0,0,0,0,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,1,1,1,1,0,1,1,1,1,0,0,1,0,1
1,0,0,0,1,1,1,1,1,1,1,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,1
1,0,0,0,0,0,0,0,0,0,2,0,1,1,1,0,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,1
1,1,1,1,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,1
1,0,0,0,0,0,1,0,2,0,0,0,1,0,1,0,0,0,3,0,0,0,1,0,1,0,1,1,1,0,1,0,1,0,1,0,0,1,0,1
1,1,1,1,1,1,1,1,1,1,0,0,1,0,1,0,3,0,0,0,1,1,1,0,1,0,0,0,0,0,1,1,1,0,1,0,0,1,0,1
1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,1
1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,1,1,1,1,1,1,1,1,1,0,0,1,0,1
1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,1,0,1
1,0,1,0,1,1,1,0,0,0,0,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,0,0,1,0,1
1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0,1,0,1
1,0,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,1,0,0,0,1
1,0,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,0,1,0,0,1
1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,0,1,0,0,1
1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,2,0,0,1,0,0,0,1,0,0,1,0,0,0,0,1
1,0,1,0,1,0,1,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,0,0,1,0,0,0,0,1
1,0,1,0,1,1,1,0,1,1,1,0,1,4,0,0,1,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,1,0,1,0,0,1
1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,0,1
1,0,1,0,1,4,0,0,0,0,1,0,1,1,1,1,1,1,1,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,0,1,1,0,1
1,0,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,4,0,0,0,0,1,0,1,0,0,1
1,0,0,0,0,0,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,0,0,1,0,3,0,1,0,0,0,0,0,1,1,1,0,0,1
1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,S,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

Para leer este fichero las funciones que se usan son las siguientes:

```
function cargarMapa(evt) {
    var fichero = document.getElementById('cargaMapa').files[0];
    if(fichero)
    {
        var leerFichero = new FileReader();
        leerFichero.onload = function(){
            var laberintoTexto = leerFichero.result;
            var dimensionesMapa =
calcularDimensionesMapa(laberintoTexto);
            laberinto = new Array(dimensionesMapa[0]);
            var filasMapa = laberintoTexto.split("\n");
            dimensionMatriz=filasMapa.length;
            for(var i=0; i<dimensionesMapa[0];i++){
                laberinto[i]= new Array(dimensionesMapa[0]);
                var columnasIMapa = filasMapa[i].split(",");
                for(var j=0; j<dimensionesMapa[1];j++){
                    laberinto[i][j]=columnasIMapa[j];
                }
            }
        }
        leerFichero.readAsText(fichero);

    }
}

function calcularDimensionesMapa(texto){
    var filas = texto.split("\n");
    var totalFilas = filas.length;
    var totalColumnas = calcularColumnas(filas[0]);
    var dimensiones = [totalFilas, totalColumnas];
    return dimensiones;
}

function calcularColumnas(fila){
    var columnas = fila.split(",");
    return columnas.length;
}
```

Las funciones **calcularDimensionesMapa** y **calcularColumnas** devuelven las dimensiones del fichero de mapa. En principio todas las estructuras de laberinto tienen las dimensiones 40x40 pero es posible utilizar mapas de otros tamaños.

La función **cargarMapa** permite, una vez que se ha seleccionado un fichero de mapa, generar una matriz laberinto con todos los valores necesarios para generar la estructura. Analicemos algunas de las instrucciones:

- **var fichero = document.getElementById('cargaMapa').files[0];**

cargaMapa es el elemento de la página HTML que viene definido por la etiqueta <INPUT type=”file”>. Cuando este botón se pulsa muestra un cuadro de diálogo desde el que debemos seleccionar un fichero. Este fichero es el alcanzado por la línea anterior.
Una vez tenemos el fichero de mapa el objeto **FileReader** es el encargado de leerlo.

- **Laberinto = new Array**

Esta línea genera el laberinto, que no es más que la matriz numérica que contiene los elementos que se incluyen en la estructura, ya sea un bloque, un espacio o un objeto.
Gracias al bucle posterior cada uno de los números almacenados en el fichero de texto pasa a la matriz laberinto que es la que se usará posteriormente para generan la escena. Leemos

línea a línea el fichero, separamos cada elemento de la línea teniendo en cuenta que la coma es el carácter delimitador y almacenamos en cada casilla de la matriz.

La función **cargarMapa** es referenciada cuando se pulsa el botón de **Seleccionar archivo**. Así, en el fichero laberinto.js la zona del código que controla el cambio del estado de este botón es esta:

```
document.getElementById("cargaMapa").onchange = function () {
    cargarMapa();
};
```

NOTA: **getElementByID("id")** es una función muy usada en javascript. Se escarga de acceder a un elemento generado con anterioridad en HTML, cuyo parámetro id sea el nombre especificado entre los paréntesis. Así en nuestro código **cargarMapa** es el **id** asignado al botón **Seleccionar archivo**.

La generación de la escena se realiza a partir de la función **crearEscena**. Veamos cada parte de esta función.

```
escena = new BABYLON.Scene(contexto);
escena.gravity = new BABYLON.Vector3(0, -9.81, 0);
escena.collisionsEnabled = true;
```

Creación de la escena sobre la que se colocarán todos los elementos. Se establecen características de gravedad y se habilita la detección de colisiones.

```
camara = new BABYLON.FreeCamera("camaraLibre", new
BABYLON.Vector3(0, 5, 0), escena);

camara.minZ = 1;
camara.ellipsoid = new BABYLON.Vector3(1, 1, 1);
camara.checkCollisions = true;
camara.applyGravity = true;
escena.activeCamera = camara;
camara.attachControl(canvas);
```

Creación de la cámara y configuración de características. Se crea una **FreeCamera** para que esta avance en función de los movimientos del usuario.

```
var materialSuelo = new BABYLON.StandardMaterial("materialSuelo",
escena);

materialSuelo.emissiveTexture  =  new  BABYLON.Texture
("texturas/suelo.jpg " , escena);

var suelo = BABYLON.Mesh.CreateGround("suelo", (mCount + 2) *
TAMA_BLOQUE, (mCount + 2) * TAMA_BLOQUE,1, escena, false);

suelo.material = materialSuelo;
suelo.checkCollisions = true;
```

Configuración del suelo. En primer lugar se crea el material que será asignado a este elemento mediante el objeto **BABYLON.StandardMaterial**, el fichero de textura para el material se asignará a través de la propiedad **emmissiveTexture**, propia de este tipo de variables.
Se crea la variable suelo a través del objeto **BABYLON.Mesh.CreateGround** estableciendo las dimensiones de este. Para obtener estas dimensiones sabemos que el laberinto estará compuesto por bloques del mismo tamaño así que se calcula la cantidad de bloques máxima en función de las dimensiones de la matriz laberinto. Si el ancho del mapa es 40 el suelo será de 42 * TAMA_BLOQUE de ahí que las dimensiones sean (mCount+2) * TAMA_BLOQUE.
Se asigna el material al suelo en la línea **suelo.material = materialSuelo**.

```
var luz = new BABYLON.DirectionalLight("luzDireccional", new
BABYLON.Vector3(0,-1,0),escena);
luz.diffuse = new BABYLON.Color3(1, 0, 0);
luz.intensity = 0.2;
```

Definición de la luz que vamos a usar en la escena. Es una luz direccional.

```
var materialCubo = new BABYLON.StandardMaterial("materialCubo",
escena);

materialCubo.emissiveTexture  =  new  BABYLON.Texture (
"texturas/ladrillos.jpg " , escena);


var cuboBase = BABYLON.Mesh.CreateBox("cuboBase", TAMA_BLOQUE,
escena);

cuboBase.material = materialCubo;
cuboBase.subMeshes = [];
cuboBase.subMeshes.push(new BABYLON.SubMesh(0, 0, 4, 0, 6,
cuboBase));

cuboBase.subMeshes.push(new BABYLON.SubMesh(1, 4, 20, 6, 30,
cuboBase));

cuboBase.rotationQuaternion =
BABYLON.Quaternion.RotationYawPitchRoll(0, -Math.PI / 2, 0);

cuboBase.checkCollisions = true;
cuboBase.setEnabled(false);
```

Para generar los muros del laberinto hemos usado cubos. En la práctica creamos un cubo base, con una textura concreta que posteriormente será clonado tantas veces como sea necesario para desarrollar todo el laberinto. En esta zona se crea en primer lugar el material del elemento y posteriormente el objeto a través de **BABYLON.Mesh.CreateBox**.
Como en los demás objetos creados se habilita la detección de colisiones.

```
var carteles =
["texturas/cartel01.jpg","texturas/cartel02.jpg","texturas/cartel03.
jpg"];

var texturaCartelUsada=0;
```

El laberinto mostrará tres carteles en algunos puntos. Los nombres de estos carteles se almacenan en un array para posteriormente ir colocándose en cubos concretos del laberinto. La variable **texturaCartelUsada** permitirá indicar de forma dinámica qué cartel colocar en cada caso.

```
var materialObj = new BABYLON.StandardMaterial("materialJugador",
escena);
materialObj.diffuseColor = new BABYLON.Color3(0, 1, 0);
materialObj.emissiveColor = new BABYLON.Color3(0, 0, 0);
materialObj.specularColor = new BABYLON.Color3(1, 1, 1);
```

El laberinto contiene objetos que sirven de referencia al jugador. Estos objetos disponen de una textura que se define aquí, justo antes de generarse la estructura.

```
for (var filas = 0; filas < mCount; filas++) {
          for (var columnas = 0; columnas < mCount; columnas++) {
                              ……..
                  }
}
```

Este es el bucle que genera el laberinto. Recorre la matriz laberinto anteriormente definida. En su interior encontraremos una sentencia condicional compuesta por las siguientes partes.

```
if (laberinto[filas][columnas]==1) {
    cubo = cuboBase.clone("nuevoCubo" + filas + columnas);
    if(columnas%25==0 && filas%5==0)
    {
       var matCartel=new BABYLON.StandardMaterial("matCartel",
escena);

       matCartel.emissiveTexture  =  new  BABYLON.Texture (
carteles[texturaCartelUsada] , escena);
       texturaCartelUsada++;
       if(texturaCartelUsada>2){
           texturaCartelUsada=0;
       }
       cubo.material = matCartel;
     }
     cubo.position = new BABYLON.Vector3(TAMA_BLOQUE / 2 + (filas -
(mCount / 2)) * TAMA_BLOQUE, TAMA_BLOQUE / 2,TAMA_BLOQUE / 2 +
(columnas - (mCount / 2)) * TAMA_BLOQUE);
}
```

Si en la posición filas, columnas del laberinto encontramos un 1 esto quiere decir que debe colocarse un cubo indicando que existe una pared en esa posición. Es por ello que se crea un objeto cubo que es clon del objeto **cuboBase** creado con anterioridad. Ahora bien, hemos indicado que algunos de estos cubos tendrán una textura diferente a la de ladrillos típica de nuestro juego, así encontramos otra sentencia condicional que cuenta cada 25 columnas y 5 filas y coloca un cartel (**if(columnas%25==0 && filas%5==0)**. Finalizamos colocando el elemento a través de la propiedad position, **cubo.position**.

```
else if(laberinto[filas][columnas]==2){
     var objCaja = BABYLON.Mesh.CreateBox("objCaja", 5.0, escena);
     objCaja.material = materialObj;
     objCaja.checkCollisions = true;
```

```
     objCaja.position = new BABYLON.Vector3(TAMA_BLOQUE / 2 + (filas
- (mCount / 2)) * TAMA_BLOQUE, TAMA_BLOQUE / 2,TAMA_BLOQUE / 2 +
(columnas - (mCount / 2)) * TAMA_BLOQUE);
}
```

Si el número encontrado en la posición filas, columnas de la matriz laberinto es un 2 significa que en esa posición debe existir un objeto de tipo cubo, así en este **else if** se genera el objeto, se le asigna el material ya creado con anterioridad y se coloca en el laberinto.

```
else if(laberinto[filas][columnas]==3){
   var objEsfera = BABYLON.Mesh.CreateSphere("sphere", 10.0, 10.0,
escena);
   objEsfera.material = materialObj;
   objEsfera.checkCollisions=true;
   objEsfera.position = new BABYLON.Vector3(TAMA_BLOQUE / 2 + (filas
- (mCount / 2)) * TAMA_BLOQUE, TAMA_BLOQUE / 2, TAMA_BLOQUE / 2 +
(columnas - (mCount / 2)) * TAMA_BLOQUE);
}
```

Este trozo de código es similar al anterior, la diferencia radica en el número que se encuentra en la matriz laberinto. Si encontramos un 3 en la casilla que estamos analizando debemos generar una esfera.

```
else if(laberinto[filas][columnas]=='E'){
   var entrada = BABYLON.Mesh.CreateBox("cajaEntrada", TAMA_BLOQUE,
escena);
   var materialEntrada = new
BABYLON.StandardMaterial("materialEntrada", escena);

   materialEntrada.emissiveTexture  =  new  BABYLON.Texture (
"texturas/entrada.png " , escena);
   entrada.material = materialEntrada;
   entrada.checkCollisions = true;

   entrada.position = new BABYLON.Vector3(TAMA_BLOQUE / 2 + (filas -
(mCount / 2)) * TAMA_BLOQUE, TAMA_BLOQUE / 2, TAMA_BLOQUE / 2 +
(columnas - (mCount / 2)) * TAMA_BLOQUE);

   var x = TAMA_BLOQUE / 2 + (filas - (mCount / 2)) * TAMA_BLOQUE;
   var y = TAMA_BLOQUE / 2 + ((columnas-2) - (mCount / 2)) *
TAMA_BLOQUE;
   camara.position = new BABYLON.Vector3(x, 5, y);
}
```

Si encontramos una E en la casilla analizada del laberinto esto significará que estamos en la casilla de salida. Este cubo tiene una textura especial al resto ya que se muestra con fondo blanco y una letra E en la parte d¡central.
El cubo se colocará como el resto justo después de la asignación del material.
Como el jugador debe comenzar su andadura en la casilla de entrada al laberinto se configuran las coordenadas iniciales de la cámara justo en esta posición. Por ello creamos dos variables **x** e **y** que detectan estos valores que serán posteriormente asignados a **camaro.position**.

```
else if(laberinto[filas][columnas]=='S'){
    var salida = BABYLON.Mesh.CreateBox("cajaSalida", TAMA_BLOQUE,
escena);

    var materialSalida = new
BABYLON.StandardMaterial("materialSalida", escena);
```

```
    materialSalida.emissiveTexture  =  new  BABYLON.Texture (
"texturas/salida.png " , escena);

    salida.material = materialSalida;
    salida.checkCollisions = true;

    salida.position = new BABYLON.Vector3(TAMA_BLOQUE / 2 + (filas -
(mCount / 2)) * TAMA_BLOQUE, TAMA_BLOQUE / 2, TAMA_BLOQUE / 2 +
(columnas - (mCount / 2)) * TAMA_BLOQUE);

    coord_x_Salida = salida.position.x;
    coord_y_Salida = salida.position.y;
```

Similar al **else if** anterior. Se encarga de señalizar la posición de la salida del laberinto que se visualiza como un cubo de fondo blanco y letra S en el centro.
Las variables **coord_x_Salida** y **coord_y_Salida** almacenan la posición **x** e **y** de la salida para ser usadas en la parada del cronómetro en caso de alcanzar esta posición.

```
jugador = BABYLON.Mesh.CreateBox("box", 4.0, escena);
var materialJugado = new BABYLON.StandardMaterial("materialJugador",
escena);
materialJugado.diffuseColor = new BABYLON.Color3(1, 0, 0);
materialJugado.emissiveColor = new BABYLON.Color3(1, 0.2, 0.2);
materialJugado.specularColor = new BABYLON.Color3(1, 1, 1);
jugador.material = materialJugado;
jugador.position = camara.position;
```

La posición del jugador es almacenada por si desea realizar una vista aérea del laberinto. Así el jugador se representa mediante un cubo de color rojo que es colocado en la posición de la cámara. Al comenzar el juego el jugador se encuentra en la entrada del laberinto.

```
return escena;
```

Para finalizar, la escena generada incluyendo todos los elementos vistos es devuelta para que se muestre en el **canvas**.

La escena se crea al comenzar el juego, es por ello que veremos en el código las siguientes líneas:

```
document.getElementById("inicio").onclick = function () {
    canvas = document.getElementById("canvas");
    contexto = new BABYLON.Engine(canvas, true);

    horas = 0;
    minutos = 0;
    segundos = 0;
    cronometro();

    window.addEventListener("resize", function () {
        contexto.resize();
    });

    entorno = crearEscena();
```

```
    entorno.activeCamera.attachControl(canvas);

    contexto.runRenderLoop(function () {
        if(camara.position.x <= (coord_x_Salida + TAMA_BLOQUE) ||
camara.position.y <= (coord_y_Salida + TAMA_BLOQUE)){
                alert("Tiempo transcurrido para la finalización del
recorrido: " + horas + ":" + minutos + ":" + segundos);
        }
        entorno.render();
    });
};
```

La función se asocia al evento clic del botón **Comenzar juego** cuyo **id** es **inicio**. Vemos como se realiza la llamada al método **crearEsccena**. **Contexto.runRenderLoop** representa la función que se llama de forma continuada y que permite visualizar el laberinto tras los diferentes movimientos del jugador. Justo aquí, si llegamos a la posición de salida haremos que se muestre un mensaje indicando el tiempo transcurrido en realizar el recorrido (**if(cámara.position.x <= …**).

La función **cronometro** se encarga de iniciar un cronómetro justo en el momento en que el jugador tras cargar el mapa seleccionado inicia el juego. A continuación se muestra su contenido. Es una función javascript recursiva de forma que cada segundo vuelve a llamarse y cargar las variables horas, minutos y segundos en función de sus valores anteriores.

```
function cronometro(){
    var t_Horas;
    var t_Minutos;
    var t_Segundos;

    if (segundos < 59){
        segundos = segundos + 1;
    }
    else{
        segundos = 0;
        if(minutos < 59)
            minutos = minutos + 1;
        else{
            minutos = 0;
            if (horas < 24){
                horas = horas + 1;
            }
            else{
                horas = 0;
            }
        }
    }

    t_Horas = horas;
    t_Minutos = minutos;
    t_Segundos = segundos;

    if (horas < 10) {t_Horas = '0' + t_Horas;}
    if (minutos < 10) {t_Minutos = '0' + t_Minutos;}
    if (segundos < 10) {t_Segundos = '0' + t_Segundos;}

    document.getElementById("reloj").innerHTML = t_Horas+':'+
t_Minutos+':'+ t_Segundos;
```

```
    setTimeout(cronometro, 1000);

}
```

Para conseguir una vista aérea del laberinto usamos la función **cambioVistaAerea**. Esta función básicamente se encarga de generar una nueva cámara a la que se cambia la posición y rotación para que toda la estructura pueda observarse desde arriba. La función se llama desde el botón **Vista superior/vista normal** siempre que estemos viendo el laberinto desde dentro, en vista normal.

```
var cambioVistaAerea = function (posLaberinto, posAerea,
                                                    rotacionLaberinto,
rotacionAerea) {

    var camaraAerea = new BABYLON.Animation("camaraAerea",
"position", 30, BABYLON.Animation.ANIMATIONTYPE_VECTOR3,
BABYLON.Animation.ANIMATIONLOOPMODE_CYCLE);

    var posiciones = [];
    posiciones.push({
        frame: 0,
        value: posLaberinto
    });
    posiciones.push({
        frame: 100,
        value: posAerea
    });

    camaraAerea.setKeys(posiciones);

    var camaraAereaRotacion = new
BABYLON.Animation("camaraAereaRotacion", "rotation", 30,
BABYLON.Animation.ANIMATIONTYPE_VECTOR3,
BABYLON.Animation.ANIMATIONLOOPMODE_CYCLE);

    var rotaciones = [];
    rotaciones.push({
        frame: 0,
        value: rotacionLaberinto
    });
    rotaciones.push({
        frame: 100,
        value: rotacionAerea
    });

    camaraAereaRotacion.setKeys(rotaciones);
    camara.animations.push(camaraAerea);
    camara.animations.push(camaraAereaRotacion);

    escena.beginAnimation(camara, 0, 100, false);
};
```

```
document.getElementById("vistaSuperior").onclick = function (){
    if (!vistaAerea) {
            vistaAerea = true;

            posCamaraLaberinto = camara.position;
```

```
            jugador.position = camara.position;

            rotacionCamaraLaberinto = camara.rotation;
            cambioVistaAerea(camara.position,
                new BABYLON.Vector3(16, 400, 15),
                camara.rotation, new
BABYLON.Vector3(1.4912565104551518, -
1.5709696842019767,camara.rotation.z));
        }
        else {
            vistaAerea = false;
            cambioVistaAerea(camara.position, posCamaraLaberinto,
camara.rotation, rotacionCamaraLaberinto);
        }
        camara.applyGravity = !vistaAerea;
}
```