PRÁCTICA FINAL DE SISTEMAS GRÁFICOS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA GRÁFICO

JAIME FRÍAS FUNES

PRÁCTICA FINAL DE SISTEMAS GRÁFICOS

DESCRIPCIÓN

SALOON RACE consiste en un juego en el que se conducirá un coche de juguete a lo largo de un largo salón/pasillo evitando los obstáculos que se van presentando en el camino con ayuda del puntero del ratón. Conforme se avance se irá subiendo de nivel, y con esto incrementándose la dificultad (mayor número de obstáculos, velocidad, iluminación…), y además mientras se avance se irán consiguiendo puntos que se podrán canjear por powerups.

POWERUPS

Existen dos tipos de objetos para ayudar al jugador durante el juego, estos son:

- **Reloj**. Al usarlo se disminuirá la velocidad del coche para esquivar zonas con alta densidad de obstáculos. Tiene una animación en la que la pantalla pierde el color y la cámara asciende.
- **Corazón**. Incrementa un 10% de la vida del jugador.

Aparecerán después de los primeros niveles, y aparecerán más o menos en función del parámetro definido en la definición del nivel correspondiente. Además de usarse chocando contra ellos se podrán almacenar comprándose en la tienda y usándose cuando se desee.

MENÚS

Al pausar el juego se puede acceder a las opciones del juego y alterar parámetros como la iluminación, la música o la velocidad del juego, o también acceder a la tienda o elegir nivel (si se desbloquea). Cuando se abre un menú se desenfoca el juego.

PUNTOS

Al aumentar la dificultad en los niveles siguientes es necesaria una ayuda extra, esta consiste en recolectar puntos mientras se juega para después gastarlos en comprar powerups para utilizarlos cuando se desee durante el juego.

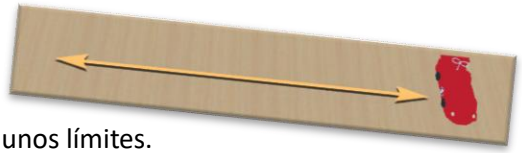
NIVELES

El juego basa su dificultad en superar 8 niveles, en cada uno de estos existirá una dificultad añadida, y para lograr esto me he basado en una definición de niveles basada en los siguientes parámetros:

- **Velocidad**. Cómo de rápido se moverá el coche en la escena.
- **Iluminación**. Las luces se pueden apagar dejando al coche sólo con los faros delanteros.
- **Número de obstáculos de cada fila.** Los obstáculos aparecen en filas, escogiéndose el número de ellos en cada fila aquí.
- **Separación entre filas de obstáculos**. Distancia entre filas de obstáculos.
- **Distancia entre powerup de salud**. Por cada cuanta distancia aparecerá un corazón para restablecer la vida.
- **Distancia entre powerup de tiempo**. Por cuanta distancia estarán separados los relojes.
- **Aceleración**. Cuanto aumentará la velocidad conforme pase el tiempo en el nivel.
- **Cámara**. Para seleccionar la posición que tendrá la cámara en cada nivel.

Esta definición de niveles se encuentra en el fichero *levels.js*, el cual declara la variable level en función de lo explicado anteriormente. Al cambiar de nivel aparece un texto en grande con el número del nivel y un texto “explicativo” del nivel en cuestión.

INTERACCIÓN / MANUAL DE USO



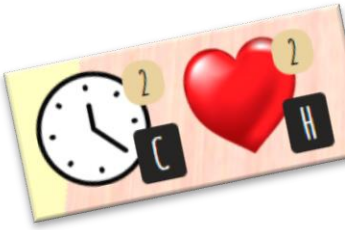
MOVIMIENTO

Se mueve el ratón para indicar al coche donde desplazarse, dentro de unos límites.

PAUSAR EL JUEGO

Al presionar la **barra espaciadora** se pausará este y aparecerá el menú.

USO DE POWERUPS



Si se compran en la tienda, se podrán usar powerups cuando se desee durante el juego.

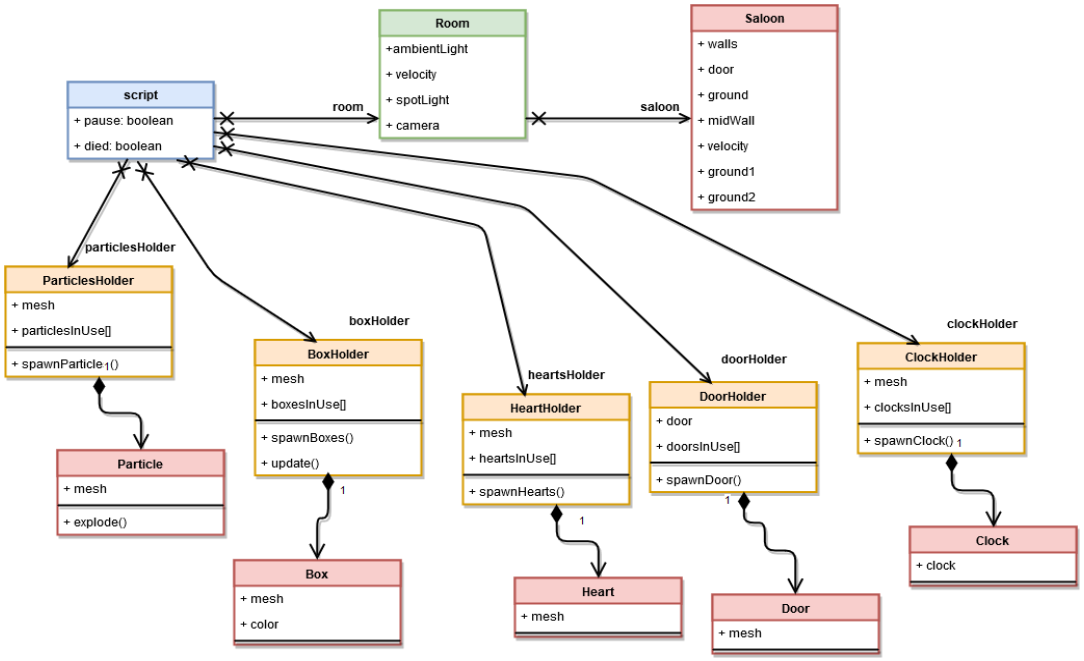
- **Ralentizar el tiempo / RELOJ.** Tecla C (*clock*).
- **Aumento de vida / CORAZÓN.** Tecla H (*heart*).

DESCRIPCIÓN DEL CÓDIGO

He estructurado las funciones del juego en diferentes archivos:

- **Script general** (script.js). Aquí se genera al render y se llama al resto de funciones, además de encontrase todo lo referido con la interacción visual del juego (menús, vida, interacción con el usuario…).
  - **Escena** (Room.js). La escena es denominada *Room*, dentro de esta clase está la mayoría de lo que compone a esta, la cámara, luces…
    - **Saloon** (Saloon.js). Clase que engloba al suelo, paredes…
  - **Movimiento** (motion.js). Aquí se encuentra la generación de objetos interactivos y en masa. Utilizo vectores de objetos y conforme se eliminan al llegar al límite de la habitación o al chocar con ellos, generándose continuamente nuevos.
  - **Niveles** (levels.js). Definición de los niveles a superar, con sus respectivos parámetros. Se modifica la variable global *level* al alternar entre un nivel u otro.

- **Índice HTML** (index.html). Fichero HTML donde se definen los elementos HTML del juego y se enlaza con el script general, con las librerías y los estilos.



Con respecto a la generación de objetos he diferenciado entre objetos “estáticos” y objetos interactivos:

- **Objetos estáticos** (Clase Saloon). No se renuevan, son los mismos objetos que aparecen continuamente al recorrer el pasillo y el jugador no interactúa con ellos.
    - **Suelo**. Una dificultad añadida fue la de que cada nivel tuviera un suelo diferente, esto lo conseguí intercalando continuamente dos suelos y cuando el nivel se viera modificado, el suelo siguiente ya sería con la nueva textura y colocado inmediatamente después del otro suelo.
    - **Paredes**. Son estáticas, son el único objeto que no se desplaza ni cambia.
- **Objetos interactivos** (archivo *motion.js*). Estos aparecen y se borran o al chocar contra el jugador o al llegar al límite de la escena. Se almacenan en vectores en los que continuamente al borrarse se crean nuevos.
    - **Partículas**. Son pequeños tetraedros que se generan en masa cuando se invoca a su función explotar. Esta explosión será del color del objeto colisionado y su distancia de explosión es variable.
    - **Obstáculos**. Pueden ser circulares, rectangulares o cónicos, y su color y tamaño son aleatorios. Al chocarse con el jugador explotan en objetos *Partícula*.
    - **Corazones**. Son shapes con extrusión que giran para diferenciarse de los obstáculos, al colisionar dan vida al jugador.
    - **Relojes**. Son figuras que giran y aparecen poco que ralentizaran la velocidad temporalmente. Cuando se colisiona además se crea un efecto de cambio de saturación y de cinemática.
    - **Puertas**. Estas no interactúan con el jugador, pero sí que tienen la misma mecánica que los objetos descritos anteriormente. Se pueden generar aleatoriamente tanto a la izquierda como a la derecha.

LIBRERÍAS

Además del propio THREE.js, he utilizado **MTLloader** y **OBJLoader** para cargar objetos, **TweenMax** para la animación de las partículas en las colisiones, **FontAwesome** para los iconos en la GUI y **JQuery**.

DISEÑO INTERFAZ

El diseño de la GUI y de las imágenes es propio. He utilizado Photoshop tanto para el logo como la barra de vida y luego gracias a *JQuery* he conseguido transiciones entre textos y ventanas, o efectos de blur al pausar el juego, y también de saturación al usar los relojes. Al clicar sobre los botones hay además sonido.

REFERENCIAS

Al tener problemas con la generación y borrado de objetos en masa continuamente busqué alguna forma eficiente de hacerlo, encontré un tutorial que resolvía el mismo problema aplicado a otro juego y aprendí de él y lo apliqué a mi generación de objetos modificándolo para adaptarlo a mi problema. Tutorial en cuestión de tympanus.net gracias a Karim Maaloul.

CAPTURAS