

PRÁCTICAS DE SISTEMAS GRÁFICOS

DISEÑO Y DESARROLLO DE UN SISTEMA GRÁFICO SOBRE UN JUEGO DE CARRERAS



Índice

Plantilla para la descripción de vuestro juego.....	3
NOMBRE DEL JUEGO.....	3
Personas que forman el grupo de prácticas.....	3
Descripción del diseño.....	3
Descripción de la implementación.....	7
1. Implementación de la Escena Principal.....	7
Círculo.....	7
ESCENA PRINCIPAL.....	8
Diagrama de clases.....	9
2. Implementación del Personaje.....	10
Modelo jerárquico.....	10
Movilidad.....	11
Colisiones y Picking.....	11
3. Implementación de los obstáculos.....	12
Conos de tráfico.....	12
Neumáticos.....	12
Zonas de turbulencias.....	12
4. Implementación de los premios.....	13
Rampa.....	13
Impulsor.....	13
Escudo.....	13
Punto de Energía.....	13
Objetos voladores (Monedas).....	13
Moneda básica.....	13
Moneda Premium.....	14
5. Interfaz.....	14
6. MANUAL DE USUARIO.....	15
CONTROLES.....	15

PLANTILLA PARA LA DESCRIPCIÓN DE VUESTRO JUEGO**NOMBRE DEL JUEGO****SWAD Speedster****PERSONAS QUE FORMAN EL GRUPO DE PRÁCTICAS**

Nombre del/de la Estudiante: **ÁLVARO RUIZ LUZÓN**

Nombre del/de la Estudiante: **ADRIÁN ROMERO VÍLCHEZ**

DESCRIPCIÓN DEL DISEÑO

SWAD Speedster será un juego de carreras donde habrá varios objetos 3D hechos mediante diferentes técnicas de modelado.

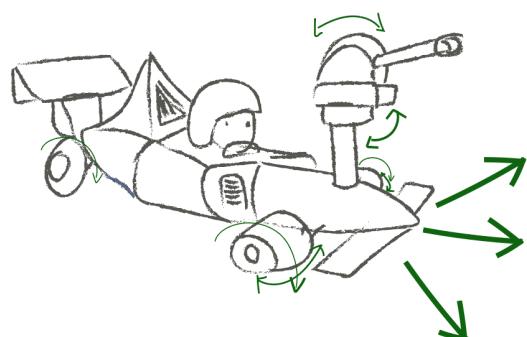
• Personaje principal

El personaje será un piloto de Fórmula 1 que irá en un monoplaza.

Además, el vehículo dispondrá de un cañón en la parte delantera, desde la que podremos disparar a las monedas que nos vayamos encontrando por el circuito.

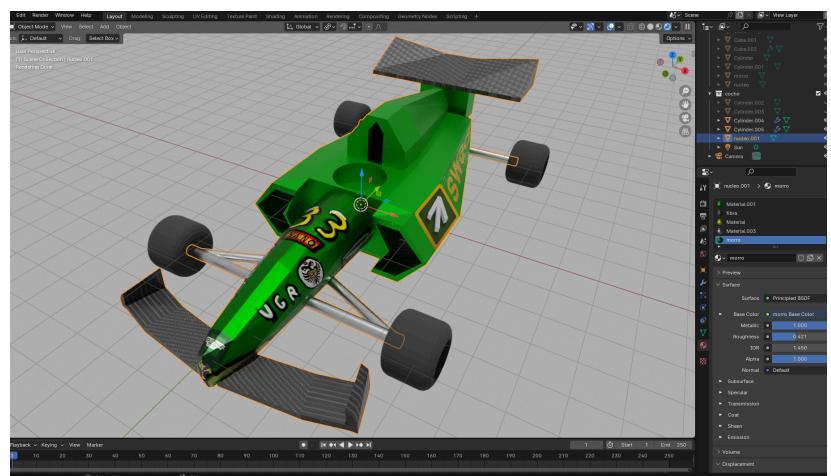
Todo esto será el objeto articulado, y tendrá 3 grados de libertad. El primero, será el que esté en continuo movimiento, y serán las ruedas del vehículo. Además, las ruedas delanteras tendrán un movimiento de rotación acotado de entre -45 y 45 grados, para que se muevan de izquierda a derecha.

El segundo grado de libertad será un movimiento acotado del cañón sobre el eje Y. Y el tercer grado de libertad será un movimiento acotado del cañón sobre el X.



En el boceto se puede apreciar más o menos todo lo anterior explicado:

El chasis del monoplaza está diseñado en Blender por el alumno Álvaro y la idea es exportarlo al proyecto como .OBJ y con texturas (como se ve en la imagen: el logo de SWAD y UGR). Sólo se hará la base con OBJ, la idea es añadir aparte las ruedas y el piloto.



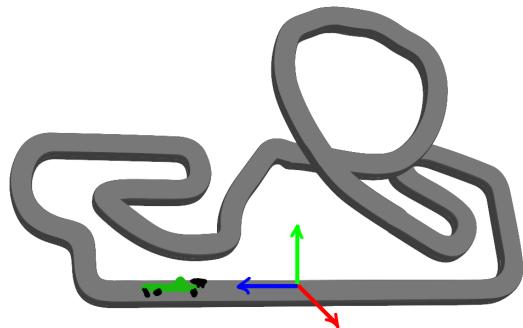
• Elementos

Los elementos que formarán parte del proyecto junto al personaje principal serán los siguientes:



a. Circuito

El circuito será un TubeGeometry de THREE cerrado, donde el personaje principal irá recorriendo alrededor de su radio y dando vueltas. Como inspiración usaremos el circuito de Montmeló pero añadiéndole alguna cosa como una pirueta.



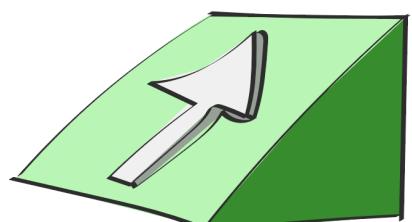
En la imagen se ve un boceto aproximado:

b. Obstáculos

i. Buenos

- **Rampas:** Pequeñas rampas que brindarán un pequeño impulso al jugador.

Cuando el jugador colisione con la rampa, éste irá trasladando verticalmente hacia arriba y con una velocidad mayor (x1.5 por ej) mientras que esté en el “aire”. Cuando haya terminado de pasar por la rampa, irá volviendo a su traslación vertical original. Posiblemente será diseñado con un Cube u otra figura que se le aplicará rotaciones y CSG.



- **Puntos de energía:** Recargarán el turbo del jugador.

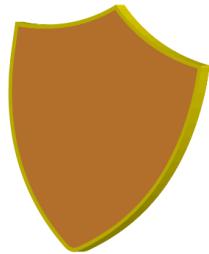
Cuando el jugador obtenga un punto de esto aumentará la velocidad al doble.

Estará diseñado mediante un Shape en forma de rayo y una esfera con materiales.



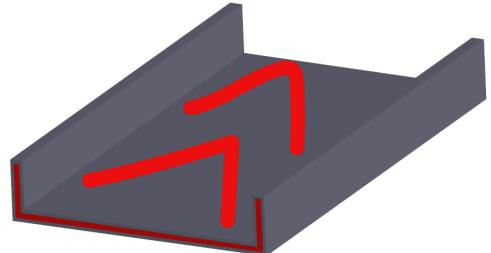
- **Puntos de escudo:** Brindan al jugador un escudo contra obstáculos y colisiones.

Se trata de un booleano que cuando esté activo, evitará que reciba **una** penalización al colisionar con un **Cono de tráfico o un Neumático**. Una vez colisione, el booleano se pondrá a false. Esto también evitará que el jugador pierda puntos. Posiblemente será diseñado mediante un Shape y extrusión.



- **Impulsores de velocidad:** Brindarán al jugador un gran impulso.

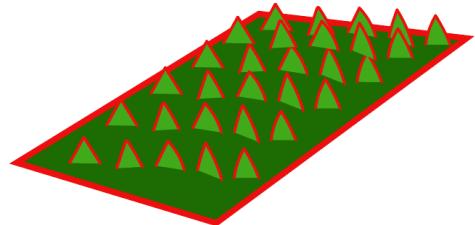
Se trata de unas placas en el suelo que, al pasar sobre ellas, el jugador recibirá un aumento en la velocidad al máximo. Posiblemente sea diseñado mediante un Geometry Box y una textura de vídeo.



ii. Malos

- **Zonas de turbulencia:** Zonas en las que si entra el jugador, su velocidad se verá reducida considerablemente.

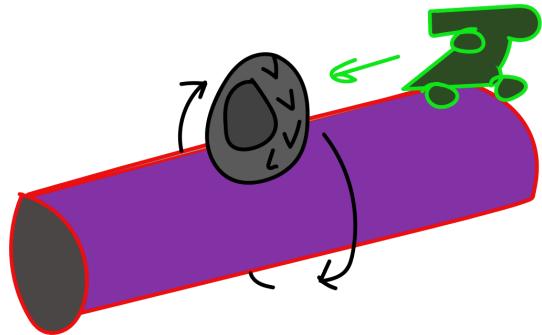
Es algo parecido a unos pinchos, que hará que cuando el jugador lo pise, pierda velocidad ($\text{velocidad} = \text{velocidad} * 0.75$ por ej). Posiblemente se construya con conos y un cubo escalado (unidos por CSG). El jugador no perderá puntos.



- **Neumáticos:** Alrededor del circuito girarán algunos neumáticos alrededor del radio del tubo que el jugador deberá esquivar.

Cada neumático será construido a partir de un perfil por revolución.

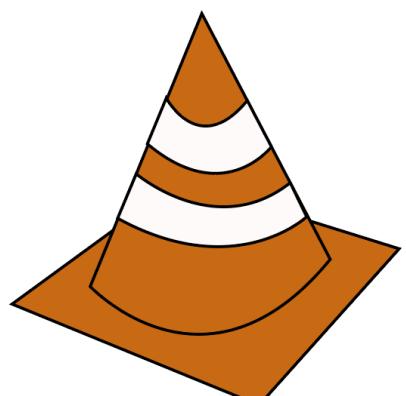
El jugador perderá velocidad al colisionar contra dicho objeto. Además, perderá una gran cantidad de puntos al chocar contra ellos.



- **Conos de tráfico:** Sobre el circuito aparecerán algunos conos, con posición fija, que el jugador deberá esquivar.

El jugador perderá velocidad al colisionar contra dicho objeto (No perderá tanta velocidad como colisionar contra un neumático). También perderá algún punto al chocar contra este obstáculo.

Posiblemente se haga con la unión de un cono y un cubo escalado.



c. Objetos voladores

- Habrá unas monedas con el logotipo de SWAD a las que se les podrá disparar con el cañón que vendrá acoplado al vehículo, y con los que iremos acumulando puntos.

El juego dispondrá de dos monedas, las normales (redondas) y las doradas (cuadradas).

**• Sistema de puntuación.**

El sistema de puntuación consiste en obtener puntos disparando a unas monedas flotantes que se encuentran alrededor del circuito. Habrá dos tipos de monedas, las normales y la moneda Premium, siendo ésta la más difícil de encontrar, acertar y, por ello, la que más puntos otorga.



El jugador dispondrá de 5 vueltas para conseguir la mayor cantidad de puntos posibles. Pero cuidado, no solo deberá estar atento a las monedas, sino también al circuito, el cual dispondrá de obstáculos que harán que el jugador pierda puntos. El circuito también dispondrá de *power ups* que aumentarán la velocidad del jugador o evitarán que pierda puntos con otro obstáculo.

Al finalizar la carrera, se obtendrá la puntuación final en función de la cantidad de monedas a las que el jugador acertó.

DESCRIPCIÓN DE LA IMPLEMENTACIÓN

1. IMPLEMENTACIÓN DE LA ESCENA PRINCIPAL

El juego se va a desarrollar en una escena donde estarán todos los objetos implementados. Trata sobre una carrera de Fórmula 1 donde nuestro piloto “Fernando Alonso” conduce su monoplaza de la escudería **AstonMartin-SWAD**, que incorpora un cañón para disparar a las monedas que hay alrededor del circuito.

El piloto deberá completar 5 vueltas para acabar la carrera mientras que consigue recoger todos los puntos posibles a través de las monedas y esquivar los obstáculos que encontrará a lo largo del circuito.

CIRCUITO

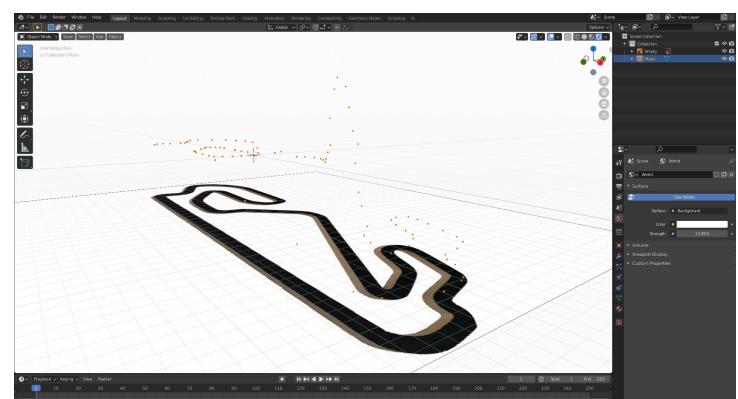
El circuito que va a recorrer el piloto es un **tubo** con la forma aproximada del circuito de Cataluña **Montmeló**, pero con alguna modificación para dar más dinamismo al circuito, como un looping/rizo y desniveles en el eje Y.

Su implementación se ha hecho de la siguiente manera:

- Diseñar los puntos que conforman el tubo: para sacar los puntos que conforman el circuito hemos tenido que usar Blender para su diseño.

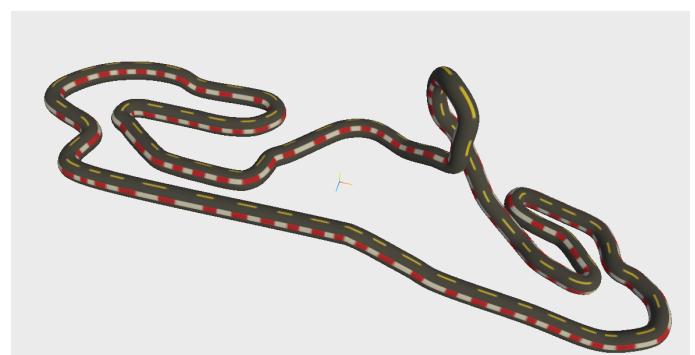
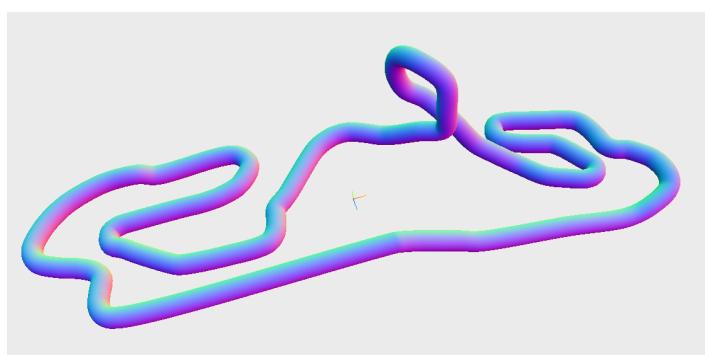
Empezamos importando la imagen de referencia en Blender, creamos un Plano y nos quedamos sólo con un vértice. Con este vértice vamos duplicando vértices siguiendo la imagen de referencia y añadiendo a mano el looping y los desniveles en el eje Y (eje Z en Blender). Una vez tengamos los puntos, lo exportamos a **.OBJ**.

- Extraer los puntos y crear el tubo: una vez tenemos el OBJ, leemos el archivo para sacar las coordenadas de cada vértice y nos las llevamos al constructor de la **clase Circuito**. Creamos una lista de THREE.Vector3 y con un proceso automático convertimos las coordenadas escritas de OBJ a THREE.js (añadir también que ha sido necesario poner a mano algún vértice extra para simular mejor el movimiento del personaje por el tubo). Esta lista de puntos se lo pasamos al TubeGeometry como CatmullCurve3.
- Agregar el material: finalmente creamos el material a partir de una textura de carretera pintada a mano en Paint y un mapa de normales para dar porosidad al asfalto.



```
import * as THREE from '../libs/three.module.js'

class Circuito extends THREE.Object3D {
  constructor(gui, titleGui) {
    super();
    var vertices = [
      new THREE.Vector3(3.513597, 0.000000, 9.059264),
      new THREE.Vector3(-36.441284, 0.000000, 8.497508),
      new THREE.Vector3(-38.001717, 0.000000, 7.473474),
      new THREE.Vector3(-38.440596, 0.000000, 5.815515),
      new THREE.Vector3(-38.148810, 0.000000, 4.108791),
      new THREE.Vector3(-37.611610, 0.000000, 1.621852),
      new THREE.Vector3(-38.489353, 0.000000, 0.287718),
      new THREE.Vector3(-40.391132, 0.000000, -0.962615),
      new THREE.Vector3(-43.749268, 0.000000, -2.288937),
      new THREE.Vector3(-46.514790, 0.000000, -3.982115),
      new THREE.Vector3(-47.220280, 0.000000, -6.409003),
      new THREE.Vector3(-45.414223, 0.000000, -9.880017),
      new THREE.Vector3(-43.188503, 0.000000, -13.250000),
      new THREE.Vector3(-39.852783, 0.000000, -16.620000),
      new THREE.Vector3(-36.517063, 0.000000, -19.990000),
      new THREE.Vector3(-33.181343, 0.000000, -23.360000),
      new THREE.Vector3(-29.845623, 0.000000, -26.730000),
      new THREE.Vector3(-26.509903, 0.000000, -30.100000),
      new THREE.Vector3(-23.174183, 0.000000, -33.470000),
      new THREE.Vector3(-19.838463, 0.000000, -36.840000),
      new THREE.Vector3(-16.502743, 0.000000, -40.210000),
      new THREE.Vector3(-13.167023, 0.000000, -43.580000),
      new THREE.Vector3(-9.831303, 0.000000, -46.950000),
      new THREE.Vector3(-6.495583, 0.000000, -50.320000),
      new THREE.Vector3(-3.159863, 0.000000, -53.690000),
      new THREE.Vector3(0.180177, 0.000000, -57.060000),
      new THREE.Vector3(3.513597, 0.000000, -9.059264)
    ];
  }
}
```



La clase Circuito es la siguiente:

· Atributos:

tubeMesh: malla del tubo.

meta: cilindro con la textura ajedrezada para el suelo de meta.

lights: array de las luces del semáforo del inicio.

· Métodos:

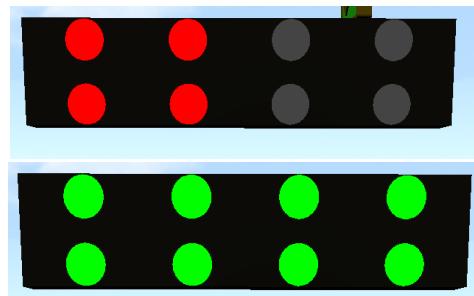
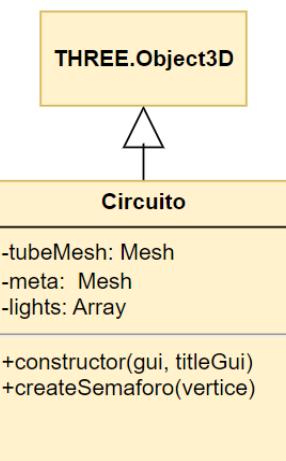
constructor(): construimos el tubo y seguidamente la meta junto a la creación del semáforo.

createSemaforo(): a partir del primer vértice del circuito posicionamos el semáforo en el inicio y creamos las luces a gris apagado. Creamos un Interval con Timeout para ir cambiando las luces a rojo hasta ponerlas todas a verde para dar comienzo a la carrera.

El circuito tiene un **semáforo** con 8 “luces” (esferas con material básico por cuestiones de rendimiento), que al principio están apagadas y al empezar el juego irán poniéndose en rojo de 2 en 2 hasta completar las 8, y seguidamente se ponen en verde, dejando ya al jugador poder moverse.

El sonido que emite el semáforo está sacado de

<https://pixabay.com/es/sound-effects/race-start-beeps-125125/>



ESCENA PRINCIPAL

La escena principal cuenta principalmente con la clase MyScene donde tiene el **renderizador**, la **luz direccional** del mundo como un sol, la creación de todos los objetos del juego, el cielo (Texture Box sacado de <https://opengameart.org/content/sky-box-sunny-day>), y el control de actualizar tanto los objetos como el HUD.

En **updateHUD()** se analiza si el jugador ha realizado una vuelta y si ha completado las 5 totales.

Además, en el mismo archivo se han creado varias funciones para el juego global:

- **alternarMusica()**: el juego cuenta con una música creada con inteligencia artificial (Suno AI) y se puede desactivar o activar desde el botón de abajo a la derecha.
- **mostrarCargando()**: es un código HTML-CSS que se muestra cuando se pulsa el **Start** del inicio y se oculta con **ocultarCargando()** una vez haya cargado la escena principal.
- **cargarEscena()**: como su nombre indica, inicializa la escena y hace el primer update para comenzar el juego.

Estas tres últimas funciones están controladas por el botón **Start del inicio**.

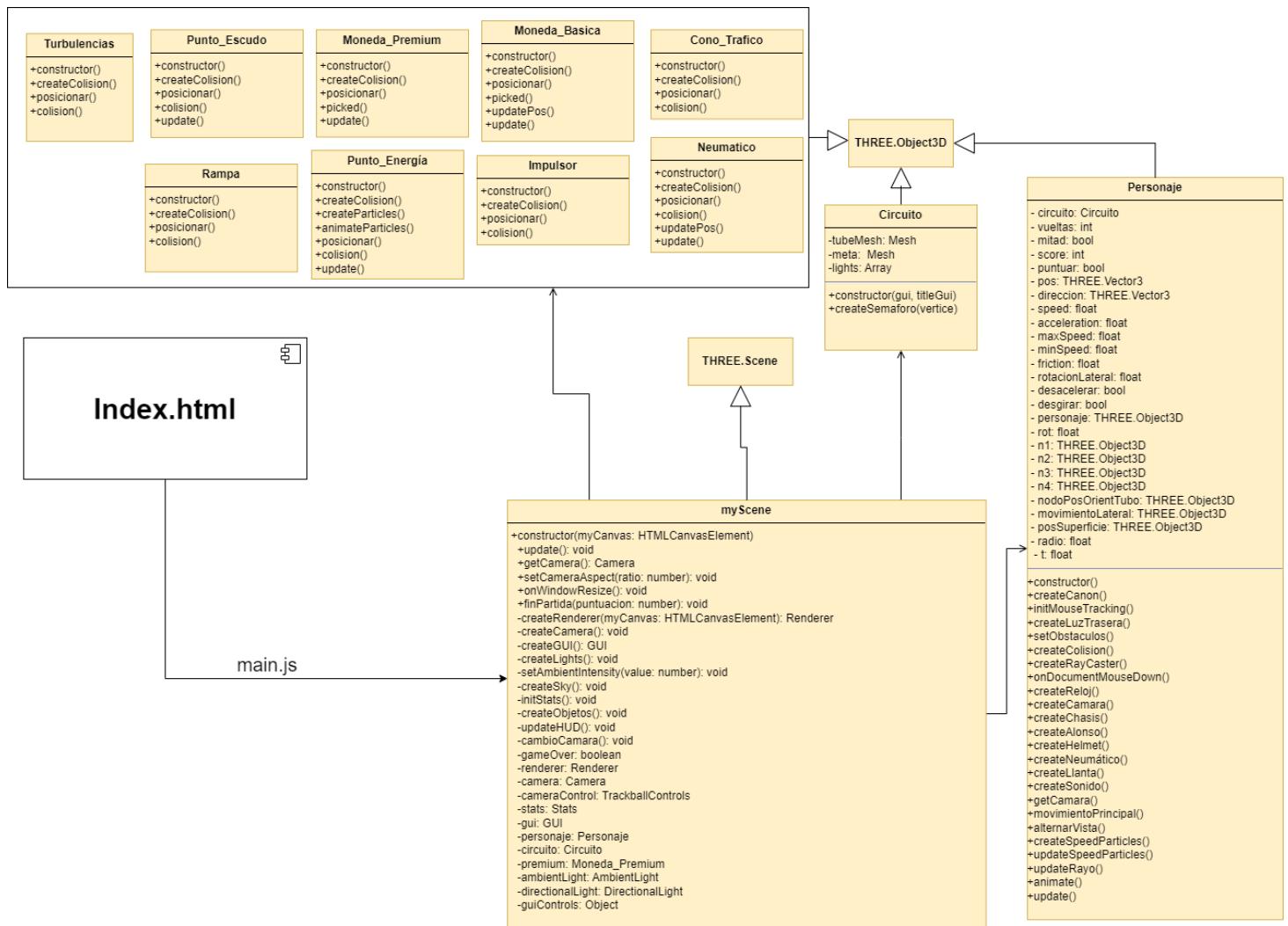
Cuando se pulsa el botón, carga la escena y comienza el juego.

Cuando se completan las 5 vueltas, el juego se para y se muestra el **fin de partida** junto a la puntuación sacada, y un botón para reiniciar la página y volver a comenzar otra carrera.



Añadir también que los sonidos restantes usados en el juego la mayoría están sacados gratuitamente de [Pixabay](https://pixabay.com).

DIAGRAMA DE CLASES:



2. IMPLEMENTACIÓN DEL PERSONAJE

MODELO JERÁRQUICO

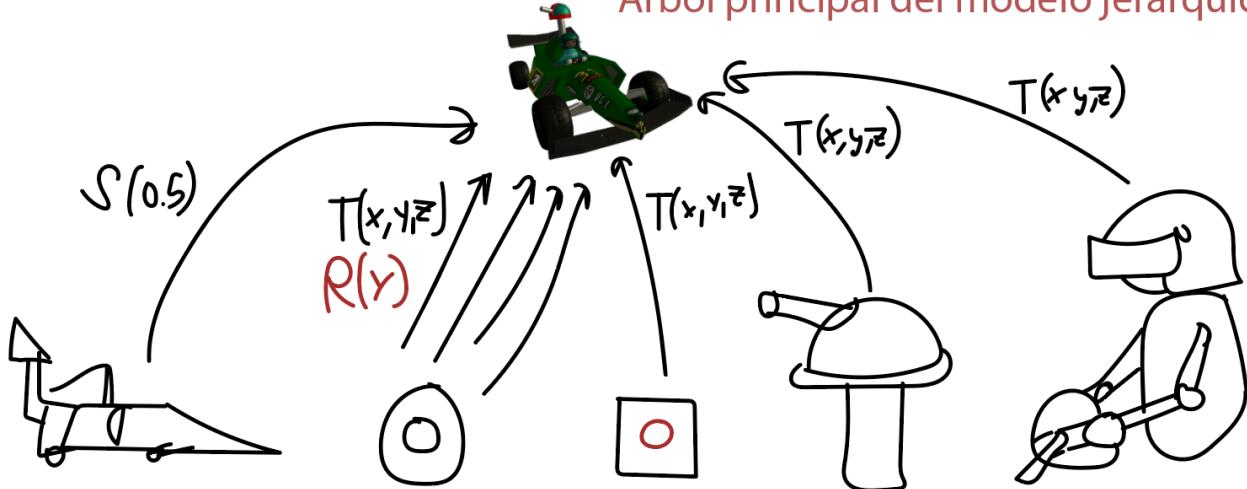
Hemos creado un coche en Blender, que hemos importado como obj para incluirlo en la escena. Luego, hemos hecho uso de una función `createCanon()` para crear un cañón de forma jerárquica, que colocaremos sobre el coche. También hemos usado una función `createNeumatico()`, que nos permite crear las 4 ruedas del modelo. Además, hemos hecho un piloto con la función `createAlonso()`, donde le hemos hecho la cabeza, el casco, el cuerpo y los brazos.

También le hemos puesto una cámara, que podrá verse en tercera persona por defecto, en primera persona si pulsamos "V", y mirará hacia atrás si mantenemos pulsado "C".

Además, le hemos puesto al coche una luz en la parte trasera que se activará cuando frenemos con la tecla "S", y si la velocidad es negativa, la luz será blanca.



Árbol principal del modelo jerárquico



Para la creación del personaje, se hacen varias funciones:

- `createChasis()`: importa el OBJ creado de Blender con sus materiales.
- `createAlonso()`: crea el piloto y el volante.
- `createHelmet()`: crea el caso para el piloto.
- `createNeumatico()`: crea un neumático con su llanta en `createLlanta()`. Se llama 4 veces a la función.
- `createCamara()`: incorporamos la cámara en tercera persona y con la tecla V se podrá alternar con primera persona y con la tecla C se mirará por detrás del personaje.
- `createReloj()`: creamos un reloj para calcular los delta entre fotogramas para el movimiento.

MOVILIDAD

La primera movilidad son las ruedas, que se mueven hacia delante o atrás dependiendo de si pulsamos **W/S**, respectivamente. También podemos rotar las ruedas con **A/D**, si la velocidad es diferente de 0. El movimiento de cañón es jerárquico, y dependerá de la posición del ratón en la pantalla, que se moverá horizontal o verticalmente si el ratón lo hace.

El movimiento no es constante, sino que dependiendo de una variable **speed** iremos incrementando o decrementando la velocidad del personaje.

El personaje cuenta con tres funciones relacionadas con la movilidad:

- **MovimientoPrincipal()**: establece las teclas para aumentar la velocidad, frenar y girar.
- **animate()**: se hacen las animaciones del personaje, como el giro de las ruedas y el movimiento del volante y brazos del piloto.
- **update()**: se aplica la velocidad a la posición del personaje y si no se está pulsando ninguna tecla se aplica un filtro que simula el rozamiento que irá parando el coche.

COLISIONES Y PICKING

El personaje necesita obtener puntos haciendo click en las monedas que se va encontrando por el circuito. Para ello usará dos funciones para el pick:

- **createRayCaster()**: crea tanto el rayo para las colisiones como el de picking.
- **onDocumentMouseEvent(event)**: activa un listener donde obtiene las coordenadas del ratón, posiciona e interseca el raycaster con las monedas disponibles. Si el rayo ha interceptado una, se selecciona y se suma la puntuación correspondiente y se llama a la función **picked()** de la moneda para hacer su animación y sonido.
- Los sonidos de las monedas están sacados de OpenGameArt: moneda básica no disponible pero está en [otros sitios web](#), [moneda Premium](#).

Para las colisiones usamos **updateRayo()**, que calcula con qué objetos del circuito (setObstáculos pasados previamente) ha intersectado con su rayo de colisión. Según el objeto, se harán ciertas animaciones, como el uso de partículas en el punto de energía y en el impulsor, la resta de puntos y reducción de velocidad por los objetos malos, o los boosts por los objetos buenos o protección con el escudo.

3. IMPLEMENTACIÓN DE LOS OBSTÁCULOS

CONOS DE TRÁFICO

El cono de tráfico consiste en un cono unido a un cubo con csg, y que se le quitará otro cono más pequeño para que la parte superior tenga un hueco. Le aplicamos una textura para mayor realismo.

Cuando el personaje choca con el cono, este cae y se queda tumbado durante 3 segundos, a la vez que hace que el personaje reduzca su velocidad un 80%. También resta un punto al jugador.



NEUMÁTICOS

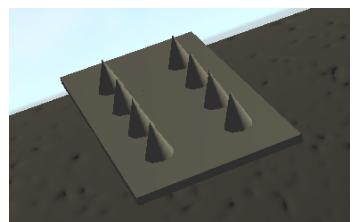
Consiste en un perfil revolucionado, al que se le ha aplicado una textura de neumático.

El neumático estará girando sobre el eje Z del tubo constantemente, y hará que al chocar con el personaje este pierda toda su velocidad. Además, hará que pierda 3 monedas. Si la cantidad de monedas es menor que 3, solo restará hasta llegar a 0, sin permitir un número negativo de monedas.



ZONAS DE TURBULENCIAS

Es un cubo rectangular al que le hemos añadido varios conos para simular unos pinchos. Cuando el personaje se desplaza sobre los pinchos, su velocidad se verá reducida a la mitad mientras el personaje se encuentra sobre el objeto, y recuperará la velocidad al pasar la zona de turbulencias.

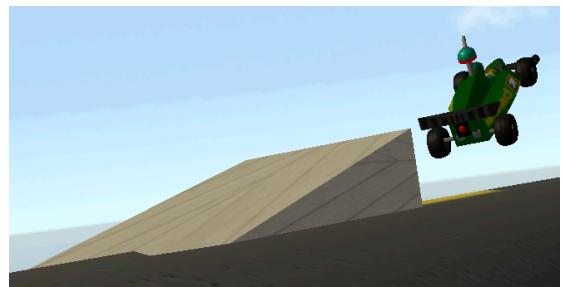


4. IMPLEMENTACIÓN DE LOS PREMIOS

RAMPA

La rampa consiste en un cubo cortado en 45 grados con otro cubo al que le hemos aplicado una textura.

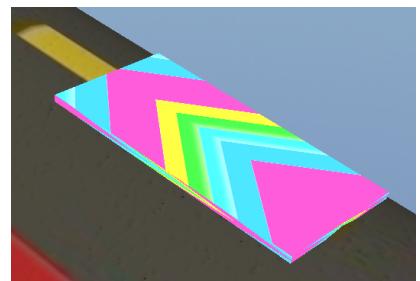
Cuando el personaje se aproxima a la rampa, aumenta su velocidad en un 50%, a la vez que incrementa la Y del personaje y rota 45 grados, simulando el levantamiento del morro del coche. Seguidamente, se vuelve a rotar -45 grados, recuperando la posición inicial del coche, a la vez que decrementa su Y.



IMPULSOR

El impulsor es solamente un rectángulo al que le hemos aplicado un video animado.

Cuando el personaje colisiona con él, alcanza su velocidad máxima durante un breve periodo de tiempo, a la vez que se crean unas partículas en pantalla dando la sensación de gran velocidad.



ESCUDO

Es un objeto3D al que le hemos añadido dos mesh, el primero es el borde del escudo, que resalta más, con un material brillante amarillo, y el segundo es el centro, con una textura de madera.

Cuando el personaje colisiona con el escudo, se activa una variable **tengoEscudo**, que nos permitirá no colisionar con un obstáculo y perder monedas o velocidad. La primera vez que el personaje entre en contacto con un obstáculo, el booleano se pondrá a **false**.



PUNTO DE ENERGÍA

Es un object3D a que le añadimos un extrude con la forma del rayo, y una esfera con la opacidad disminuida.

Cuando el personaje entra en contacto, aumenta su velocidad al doble.



OBJETOS VOLADORES (MONEDAS)

MONEDA BÁSICA

Un **shape** circular al que le abrimos un agujero, que rellenaremos con otro **shape** circular más pequeño, para darle una sensación de profundidad a la moneda. Luego, creamos otro shape con forma de flecha. Estos tres mesh los añadimos a un object3D.

Gira alrededor del eje Z del circuito, a una distancia sobre el tubo.

Darán un punto por cada moneda que se obtenga, a través del picking.

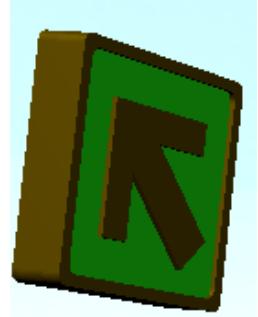


MONEDA PREMIUM

Utilizamos dos **shapes** rectangulares de la misma forma que para la moneda clásica.

Girará siguiendo una trayectoria descrita alrededor del circuito.

Dará 5 puntos por cada vez que se realice el picking sobre la moneda.

**5. INTERFAZ**

La interfaz es sencilla. Cuando ejecutamos el juego nos aparece el inicio con el botón Start y el botón de Activar/Desactivar Música abajo a la derecha:

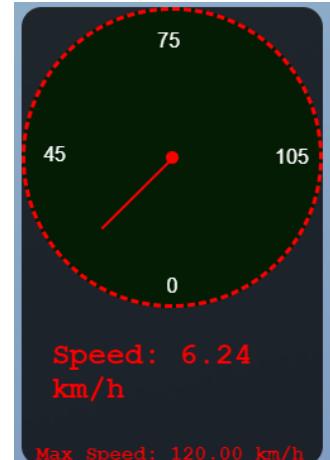


Una vez iniciada la carrera tenemos varios elementos:

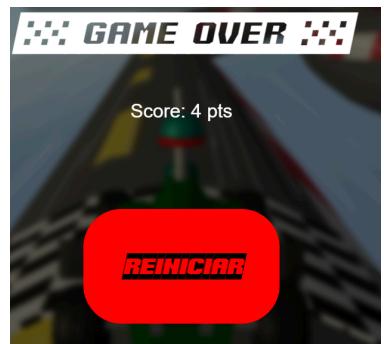
- Puntuación y vueltas:



- Cuentakilómetros (aproximadamente a la velocidad que lleva el personaje):



Y una vez acabada la carrera, saldrá el Game Over, la puntuación y el botón de reinicio:



6. MANUAL DE USUARIO

PARA INICIAR EL JUEGO, PRIMERO NECESITAMOS ACTIVAR EL SERVIDOR HTTP DENTRO DE LA CARPETA **swad-speedster/** :> `python3 -m http.server`

AHORA ABRIMOS EN EL NAVEGADOR LA DIRECCIÓN <http://localhost:8000>

Y ABRIMOS LA CARPETA **src/** Y YA SE EJECUTARÁ EL JUEGO.

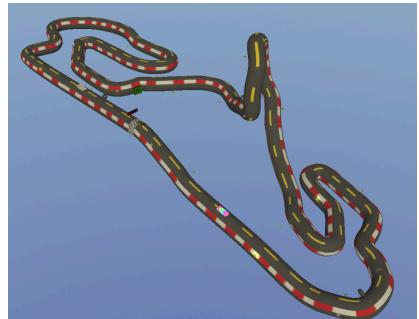
La explicación de cómo afecta cada objeto al personaje está explicado en los puntos anteriores (Implementación de obstáculos e Implementación de los premios).

CONTROLES

Para mover el personaje se hace mediante W,A,S,D y éste irá acelerando, desacelerando y girando por el circuito. Para los giros necesita velocidad, sino no gira.



Para alternar cámaras se pulsa la tecla **ESPACIO** para cambiar entre la cámara orbital y la cámara del personaje.



Cuando estemos en la cámara del personaje podemos **alternar la vista entre tercera y primera persona con la tecla V**, para así ver la animación del volante y los brazos del piloto.

También podemos mirar hacia atrás manteniendo pulsada la tecla **C**.



