Universidad de Granada

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

UNIVERSIDAD DE GRANADA

Sistemas Gráficos

Practica 2
F1 ALIEN INVASION

Profesor: Francisco Velasco Anguita
Alumnos:
Victor Casalini Gonzalez
Joaquin Salas Castillo

6 de junio de 2024

Índice

Los resultados se pueden encontrar en github a traves del siguiente enlace `https://github.com/vicmaviclu/SG`

1. Descripción del Juego

Nuestro juego se ambienta en una carrera de **Fórmula 1** que se ve interrumpida por una inesperada **invasión alienígena**.

El protagonista deberá enfrentarse a los invasores mientras continúa la carrera, disparando a *ojos voladores* y recolectando objetos beneficiosos como un *tanque de gasolina*, un *escudo F1* y un objeto *reparar* que le otorgarán puntos y mejoras diversas. Además, deberá esquivar obstáculos perjudiciales como *pinchos* y *Ovnis* para mantener su ventaja y sobrevivir a la amenaza extraterrestre.

El **Personaje Principal** es un *coche de F1* que se compone de una base, dos ejes, cuatro ruedas y un alerón. Este personaje se moverá por la superficie de nuestro circuito, pudiendo girar hacia la derecha y la izquierda si no hemos sufrido ningún daño que nos lo impida.

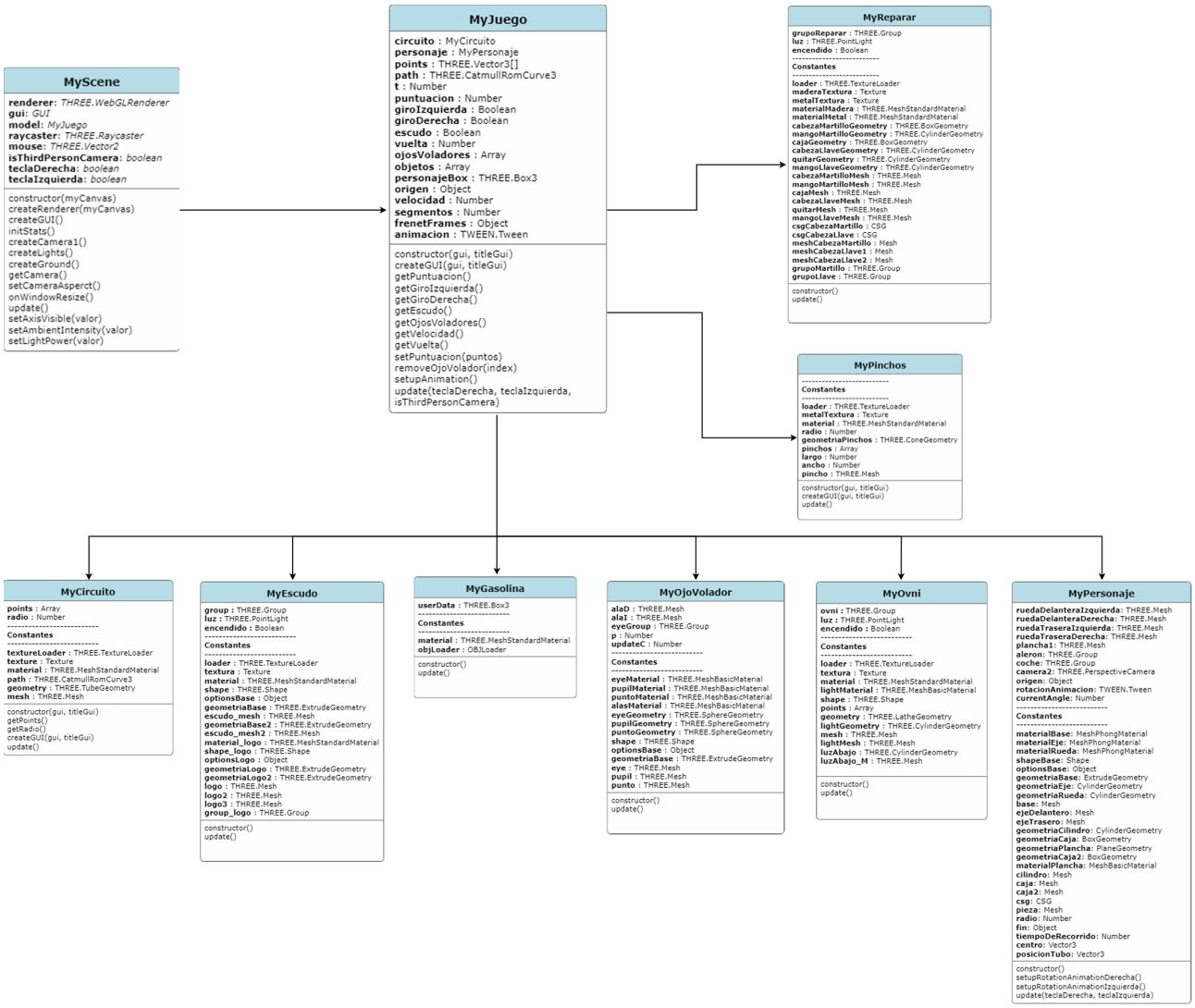
El **Objeto Articulado** es el *alerón* del coche de F1, donde la primera parte de la articulación es la totalidad de la pieza que se mueve girando sobre el eje Y. La segunda parte articulada es un rectángulo en la parte superior del alerón que rota sobre el eje x una plancha en su interior que se abre y se cierra.

Una cosa a tener en cuenta en el juego es que hemos experimentado unos errores con la animación cuando tenemos mas de *90fps*, así que se recomienda probarlo con un **máximo** de *60*.

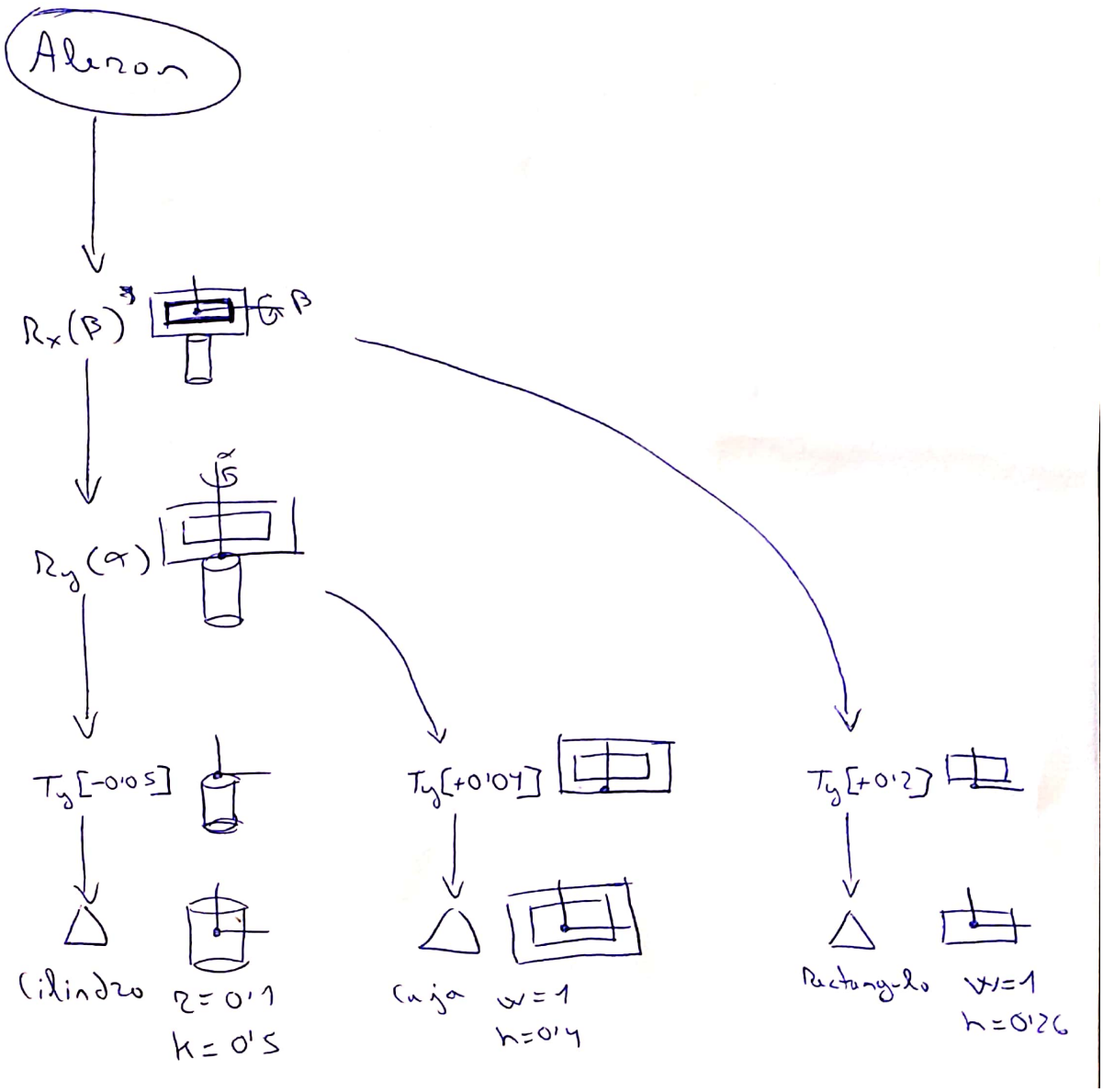
De todas formas se adjunta un vídeo en forma de demo para que se pueda comprobar el funcionamiento correcto de esta a *60fps*.

2. Diseño de la Aplicación

Diagrama de Clases



Modelo Jerárquico



Descripción de Algoritmos Importantes

- Animación

Tenemos tres animaciones en total una para el movimiento continuo del personaje por el circuito y las otras para los dos giros. Para realizar estas animaciones se ha utilizado el código de las diapositivas de teoría del uso de splines para animar una figura en el apartado de animación mediante caminos.

Así que aquí vamos a mostrar las modificaciones en nuestro código

```
// Inicializar this.origen antes de llamar a this.setupAnimation();
this.origen = { t: 0 };
this.velocidad = 0.000025;

setupAnimation() {
    ...
    var fin = { t: this.origen.t + this.velocidad };
    var tiempoDeRecorrido = 5;

    ...
    .onComplete(() => {
      if (this.origen.t > 0.999) {
          this.origen.t = 0;
          this.velocidad *= 1.10; // Aumentar la velocidad un 10%
          this.vuelta += 1;
      } else {
          this.origen.t += 0.0001;
      }
    });
    this.animacion.start();
  }
```

Listing 1: Animacion Movimiento Continuo

En este código se pueden apreciar los cambios importantes con respecto a las diapositivas. El primer cambio es configurar como variable de la clase el **origen** de la animación y el final sera el origen mas la velocidad.

El segundo cambio es que cuando se completa la animación comprobamos si ha *llegado al final del circuito*, en el caso de que esto sea correcto reiniciara los parámetros de la animación, añadirá al contador una vuelta y aumentara la velocidad de la animación.

Estos cambios son precisamente para que en vez de que la animación entera se desde el principio hasta el final, se vaya haciendo poco a poco actualizando en el método update().

```
update(){
    TWEEN.update();
    if (!this.animacion.isPlaying()) {
        this.setupAnimation();
    }
    ...
}
```

Listing 2: Actualizacion de la Animacion

Para el caso de la animación hacia los lados es igual en ambos casos solo variando la posición final del personaje por lo que vamos a explicar solo la de la derecha.

Aquí se puede ver la única diferencia que tienen ambas animaciones:

```
var fin = { angle: this.origen.angle + 2 * (Math.PI / 180) };
```

Listing 3: Fin animacion derecha

```
var fin = { angle: this.origen.angle - 2 * (Math.PI / 180) };
```

Listing 4: Fin animacion izquierda

```
this.origen = { angle: Math.PI / 2 };

setupRotationAnimationDerecha() {
    var radio = 0.46;
    var fin = { angle: this.origen.angle + 2 * (Math.PI / 180) };
    ...
    var centro = new THREE.Vector3(
        this.coche.position.x,
        this.coche.position.y - radio * Math.sin(this.origen.angle),
        this.coche.position.z - radio * Math.cos(this.origen.angle)
    );

    this.rotacionAnimacion = new TWEEN.Tween(this.origen).to(fin, tiempoDeRecorrido)
    .onUpdate(() => {
        let angle = this.origen.angle;
        this.currentAngle = angle;

        let posicionTubo = new THREE.Vector3(
            centro.x,
            centro.y + radio * Math.sin(angle),
            centro.z + radio * Math.cos(angle)
        );

        this.coche.position.copy(posicionTubo);
        this.coche.rotateX(-2 * Math.PI / 180);


    });
    this.rotacionAnimacion.start();
}
```

Listing 5: Animacion Movimiento hacia la Derecha

Como se puede observar seguimos la misma estructura que la primera animación solo que ahora definimos el origen y final con ángulos para que rote alrededor de un circulo que tiene el mismo radio que el circuito.

Primero se calcula su posición inicial en el circulo (centro) y luego a esa posición se le añade la rotación correspondiente (posicionTubo), colocando el personaje en su nueva posición y rotándolo en ese mismo ángulo sobre su eje X para que siga pegado a la superficie.

Esta animación solo esta permitida si esta en tercera persona por lo que en el update añadimos esta comprobación.

```
if(isThirdPersonCamera){
    this.personaje.update(teclaDerecha && this.giroDerecha, teclaIzquierda && this.
    giroIzquierda);
}
```

Listing 6: LLamada Animacion del Personaje

- Picking

Para poder interactuar con los objetos voladores y destruirlos para ganar puntos utilizaremos un algoritmo de picking con raycaster.

```
    this.raycaster = new THREE.Raycaster();
    this.mouse = new THREE.Vector2();
```

Añadimos un evento de *click* y solo se podrá hacer picking si estamos con la cámara en tercera persona.

```
    window.addEventListener('click', (event) => {
      // Solo se puede hacer picking si esta en tercera persona
      if (this.isThirdPersonCamera) { ... }
```

En el vector 2D que hemos definido para el ratón transformamos el sistema de coordenadas de la pantalla al sistema de coordenadas normalizado que se utiliza en gráficos 3D.

```
    this.mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
    this.mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;
```

Actualizamos el raycaster para que pase por el vector del ratón que hemos definido y comprobamos los objetos que han intersectado con el rayo, en este caso los ojos voladores.

```
    this.raycaster.setFromCamera(this.mouse, this.getCamera());
    let ojosVoladores = this.model.getOjosVoladores();
    let intersects = this.raycaster.intersectObjects(ojosVoladores, true);
```

Si el array de objetos intersectados con el rayo tiene algún objeto ojo volador entonces obtenemos el primero y vamos subiendo por los nodos padres hasta que llegamos al propio objeto MyOjoVolador. Luego buscamos el índice del ojo volador que hemos intersectado y si tenemos éxito lo eliminamos y sumamos la puntuación.

```
    if (intersects.length > 0) {
      let firstObject = intersects[0].object;
      while (firstObject.parent && !(firstObject instanceof MyOjoVolador)) {
        firstObject = firstObject.parent;
      }

      let index = ojosVoladores.indexOf(firstObject);

      if (index !== -1) {
        this.model.removeOjoVolador(index);
        this.model.setPuntuacion(5);
      }
    }
```

- Colisiones

Para las colisiones hemos usado cajas englobantes con la clase *Box3* para el personaje y los objetos.

```
    this.personajeBox = new THREE.Box3().setFromObject(this.personaje);
    for (let i = 0; i < this.objetos.length; i++) {
      let objeto = this.objetos[i];
      objeto.userData.box = new THREE.Box3().setFromObject(objeto);
    }
```

Una vez establecidas las cajas para el personaje y objetos en el método *update* actualizamos la caja del personaje para que se mueva junto a él.

```
    this.personajeBox.setFromObject(this.personaje);
```

Después comprobamos si el personaje colisiona con algún objeto con un bucle que recorre el array de objetos.

```
    for (let i = 0; i < this.objetos.length; i++) {
      let objeto = this.objetos[i];
      if (this.personajeBox.intersectsBox(objeto.userData.box)) {
      ...
      }
    }
```

Dentro del *if* comprobamos que tipo de objeto es el que ha colisionado con el personaje y en base a esto se le aplica la ventaja o desventaja correspondiente.

```
    if(objeto instanceof MyGasolina){
          this.setPuntuacion(5);
          this.velocidad *= 0.90;
        } else if(objeto instanceof MyOvni){
          if(this.escudo){
            this.escudo = false;
          } else {
            this.setPuntuacion(-5);
            this.velocidad *= 1.20;
          }
        } else if (objeto instanceof MyPinchos){
          if (this.escudo) {
            this.escudo = false;
          } else if(this.giroDerecha && this.giroIzquierda){
              if(Math.random() < 0.5){
                this.giroIzquierda = false;
              } else {
                this.giroDerecha = false;
              }
          }
        } else if (objeto instanceof MyReparar){
          if(!this.giroIzquierda || !this.giroDerecha){
            this.giroIzquierda = true;
            this.giroDerecha = true;
          }
        } else if (objeto instanceof MyEscudo){
          if (this.escudo === false) {
            this.escudo = true;
          }
        }
```

Una vez colisionado el objeto será eliminado y tras 50 segundos volverá al array de objetos para que el circuito no se quede sin objetos.

```
    objeto.visible = false;
    this.objetos.splice(i, 1);
    i--;

    setTimeout(() => {
      objeto.visible = true;
      this.objetos.push(objeto);
    }, 50000);
```

3. Manual de Usuario

El objetivo del juego es sencillo completar **5 vueltas** alrededor del circuito obteniendo el **mayor numero de puntos**.

Para cumplir este objetivo en el circuito habrá una serie de objetos con los que el jugador podrá **colisionar**, pudiendo ser esto *beneficiosos* y otorgarle alguna ventaja o *perjudicarle* y sufrir una desventaja.

También contaremos con objetos fuera de la superficie del circuito que serán los **objetos voladores**, podremos conseguir puntos de estos si los conseguimos clicar a tiempo.

En total aparecerán 32 objetos alrededor del circuito, separando su distribución en un total de 21 objetos en la superficie del circuito y 11 objetos voladores una vez colisionemos con uno de ellos este desaparecerá durante 50 segundos.

Movimientos

El movimiento de nuestro personaje será **continuo** en función a la **velocidad** que tengamos en ese momento.

Tendremos disponible dos movimientos adicionales, uno hacia la derecha pulsando la tecla flecha derecha $\rightarrow$ y otro hacia la izquierda pulsando la tecla flecha izquierda $\leftarrow$ (Estos movimientos solo están disponibles en tercera persona).

Cámaras

El juego consta de dos cámaras una **global** que muestra el circuito con todos los objetos desde fuera de este en una visión panorámica y otra en **tercera persona** sobre el personaje con la cual podemos ver lo que este tiene unos metros delante.

Para poder alternar la camara tendremos que pulsar la **barra espaciadora**.

Objetos Beneficiosos

Los objetos beneficiosos tienen un 50 % de posibilidades de aparecer.

El **tanque de gasolina** que reducirá la velocidad del personaje haciendo mas fácil evitar otros obstáculos y disparar a los objetos voladores y le sumará 3 puntos. La probabilidad que tiene de aparecer en el circuito es de *15 %*

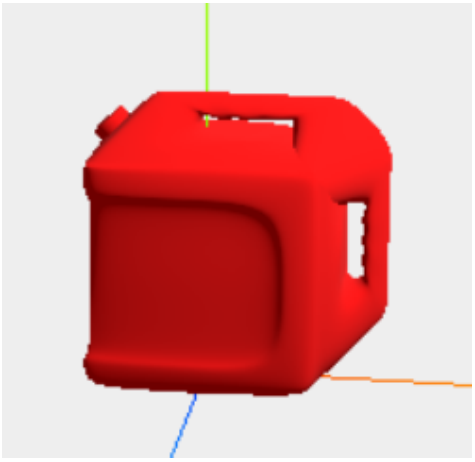


Figura 1: Tanque de Gasolina

El **Escudo F1** que protegerá al personaje, permitiéndole evitar el efecto negativo de un objeto perjudicial al chocar con él una vez. La probabilidad que tiene de aparecer en el circuito es de *45 %*

Figura 2: Escudo

Por último, el objeto **Reparar** que es una llave inglesa y un martillo que le permitirá arreglar los daños provocados por lo pinchos. En el caso de tener una rueda pinchada la reparara desbloqueando otra vez el giro en esa dirección. La

probabilidad que tiene de aparecer en el circuito es de *40 %*

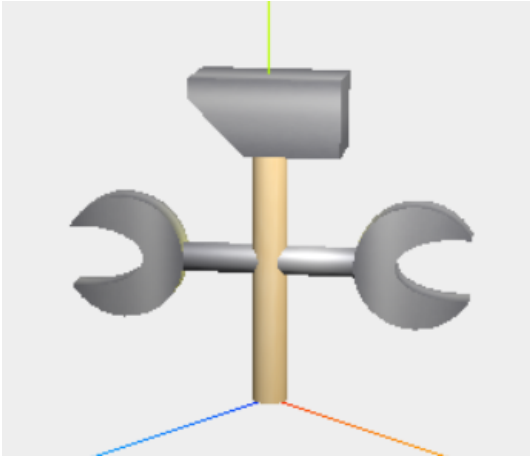


Figura 3: Objeto Reparar

Objetos Perjudiciales

Los objetos perjudiciales tienen un 50 % de posibilidades de aparecer.

Los **pinchos** pincharan una rueda aleatoriamente y le quitarán al jugador la capacidad de girar a izquierda o derecha hasta que se recoja el objeto de reparar. La probabilidad que tiene de aparecer en el circuito es de *50 %*

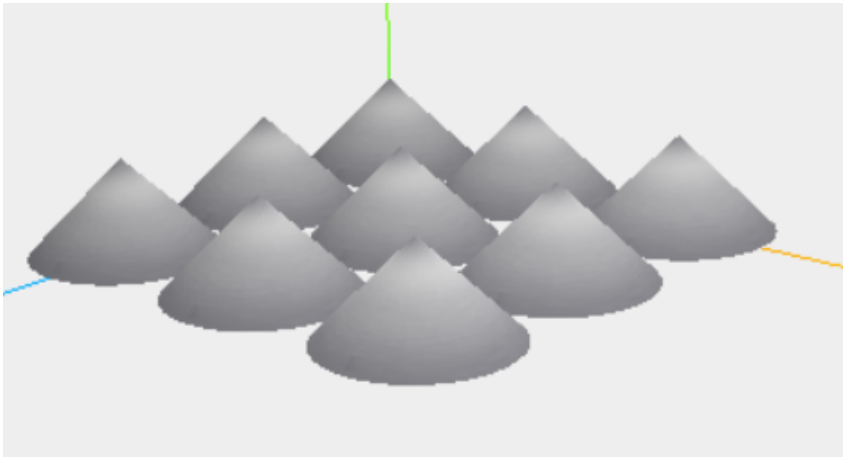


Figura 4: Pinchos

Los **Ovnis** harán perder 5 puntos al jugador y también aumentaran su velocidad lo que hará mas difícil conseguir y esquivar otros objetos.

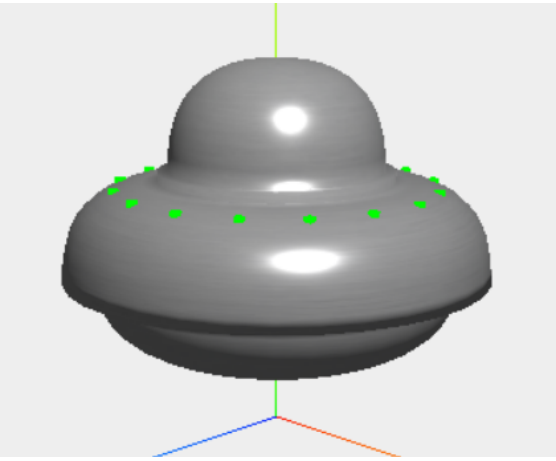


Figura 5: Ovni

Objetos Voladores

Habrá un total de once objetos voladores colocados alrededor del circuito, si conseguimos clicar en alguno **(solo valido modo tercera persona)** este desaparecerá, volviendo a reaparecer una vez han pasado 50 segundos.

Repartidos por todo el espacio habrá **Ojos con Alas** a los que el jugador podrá hacer picking (clicar) para ganar 5 puntos.

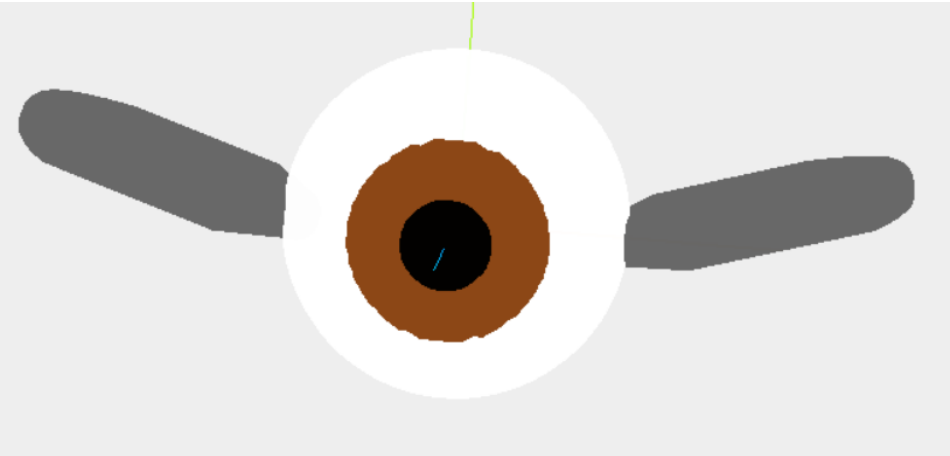


Figura 6: Ojos Voladores

Interfaz Usuario

El juego cuenta con una interfaz donde podemos ir comprobando el estado de nuestra partida.

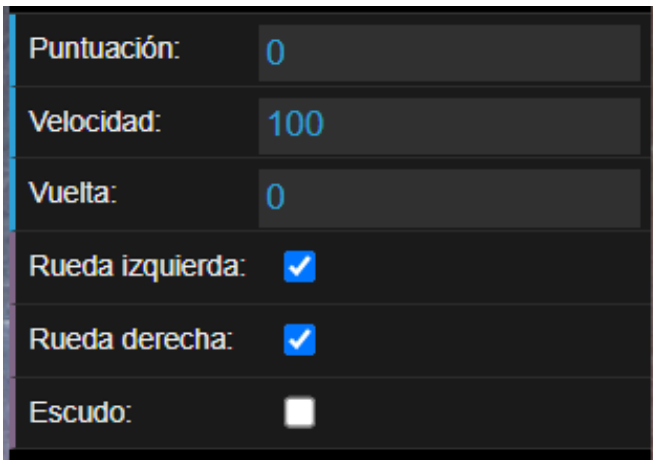


Figura 7: Interfaz con Valores Iniciales

En ella se pueden comprobar los valores de **puntuación** saber cuantos puntos hemos conseguido hasta el momento, **velocidad** con ella podemos comprobar cuanto ha variado nuestra velocidad en porcentaje con respecto a la inicial, **vuelta** podremos saber cuantas vueltas nos quedan hasta el final mediante este contador.

También dispones de unos checks que nos indican el estado de **rueda izquierda** y **rueda derecha** que indican si hemos pinchado o no alguna rueda y por tanto podemos o no girar en ese sentido, y por ultimo el **escudo** nos indica si tenemos disponible o no el objeto escudo.

Circuito

Este es el circuito que esta disponible en nuestro juego. El comienzo y final de este es el eje de coordenadas.

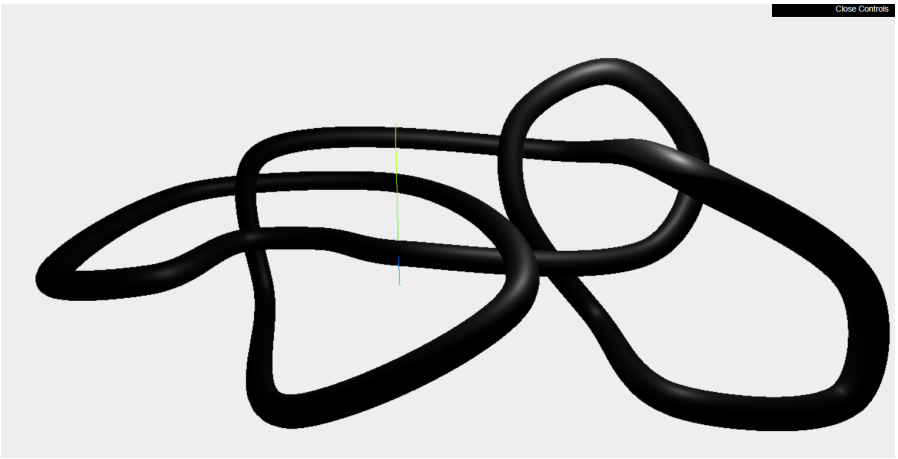


Figura 8: Circuito

4. Referencias a Modelos y Materiales Descargados

- El tanque de gasolina lo hemos descargado de: https://free3d.com/3d-model/gas-can-54646.html
- La textura de metal empleada para el objeto Reparar se ha descargado de: https://www.freepik.es/foto-gratis/fondo-plateado-metalizado-texturado_4246716.htm
- La textura para el canal de relieve que simula unas grietas ha sido descargada de: https://pngtree.com/freepng/cracked-ground-texture_5439981.html
- La textura de aluminio para el objeto Ovni la hemos descargado de: https://www.istockphoto.com/es/foto/metal-cepillado-textura-gm183803298-16144316
- La textura de que da el aspecto de asfalto al circuito se ha descargado de: https://www.textures.com/download/3DScans0604/138015
- La textura del fondo de la escena que simula una galaxia la hemos descargado de: https://unsplash.com/es/fotos/galaxy-digital-wallpaper-rCbdp8VCYhQ