# Flag Hunt: Diseño de la apliación

Iván Gijón
Víctor Mojica

2 de junio de 2024


UNIVERSIDAD
DE GRANADA

# Índice

# 1. Descripción del juego

El objetivo principal del juego será guiar un tanque por el circuito mientras recoge banderas de países.

Cuantas más banderas obtenga más puntuación obtendrá y además tendrá que ir evitando los diferentes obstáculos del circuito, que dificultarán la tarea de obtención de banderas y penalizarán al personaje cada vez que colisione con un obstáculo.

El tanque consta de dos cameras intercambiables y todos los objetos del juego tienen un material único (color o textura).

La escena cuenta con una luz ambiebtal de poca intensisdad para dar algo de luminosidad. Aparte de esta luz el resto de objetos tienen varias luces que facilitan la visión y la interaccion con el usuario. Para hacerlo mas atractivo visualmente.

El circuito es cerrado sacado de la geometria de un Torus Knot y adaptado para el correcto funcionamiento del juego. Por cada vuelta al circuito la velocidad se incrementa un 10

La idea es obtener el máximo numero de puntos sin perder las 100 vidas que se tienen en el comienzo del juego.

# 2. Modelado De Objetos

En esta sección, se muestran los objetos desarrollados y sus funcionalidades, destacando las mejoras y adiciones respecto a la propuesta anterior. Cada objeto se describe brevemente en términos de su creación y su utilidad final en el juego.

## 2.1. Objetos Modelados

- **Tanque**
    - *Descripción:* El tanque es el objeto principal de nuestra práctica. Inicialmente, se modeló utilizando CSG, lo cual impedía su

animación. Aunque visualmente no se ha modificado, hemos cambiado su estructura para permitir la animación. CSG, lo que no permitía su animación.

- *Modelo Jerárquico:* El diseño del tanque se basa en un modelo jerárquico que incluye dos grados de libertad obligatorios: la rotación de la cabeza del tanque y el movimiento del cañón.

- La figura a continuación muestra el modelo jerárquico del tanque:

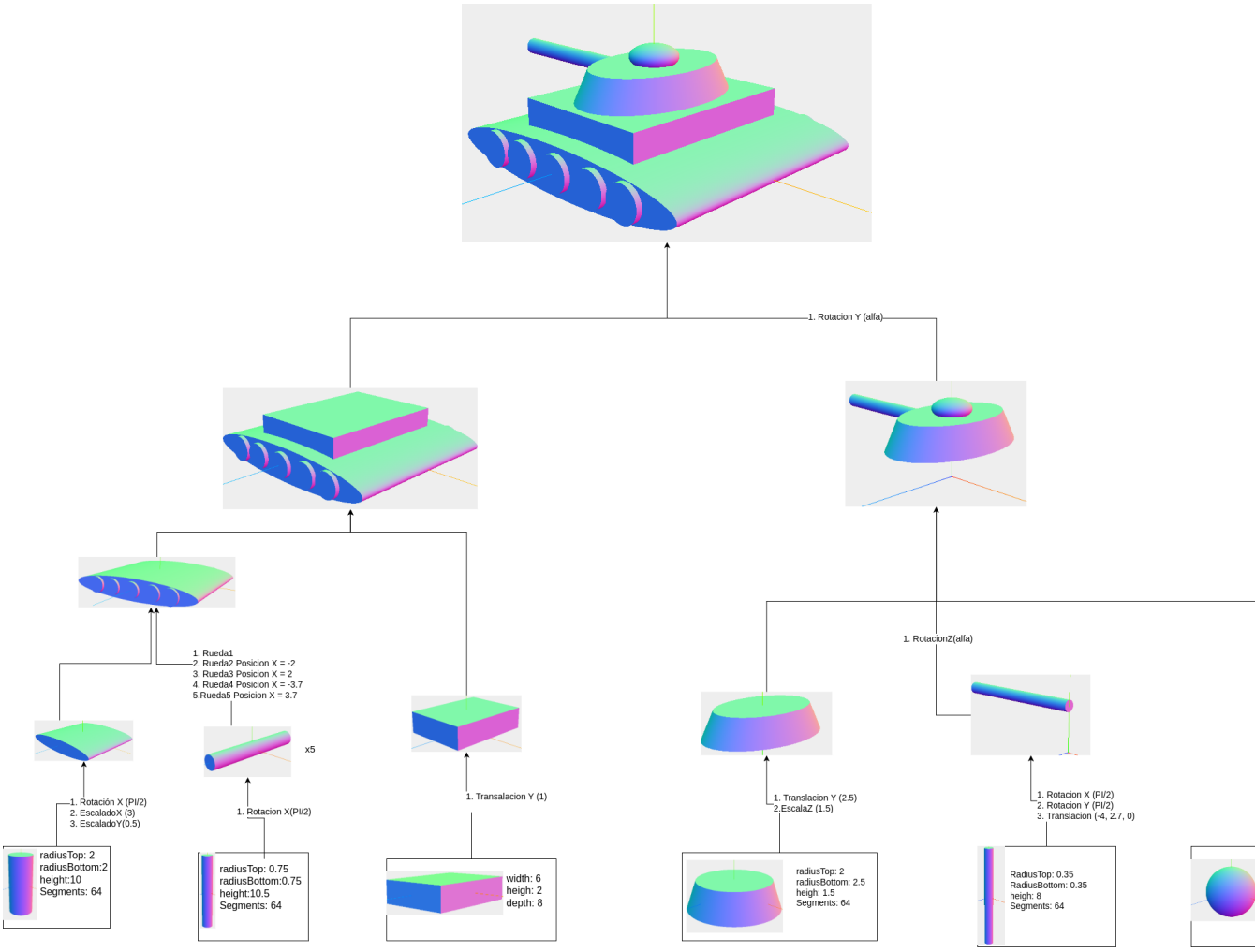


Figura 1: Modelo Jerárquico del tanque
Enlace a una vista completa que permite zoom

- *Textura:* Aplicamos una textura de camuflaje al tanque utilizando una imagen básica sin atributos adicionales. Enlace a la textura: `https://3dtextures.me/?s=cammo&orderby=relevance&order=DESC` La siguiente figura muestra el tanque con la textura aplicada:

Figura 2: Tanque con la textura aplicada

- **Botiquín**
  - *Descripción:* El botiquín se modeló utilizando una extrusión para la base, con bordes y esquinas redondeadas. La cruz en el botiquín se compone de dos rectángulos unidos mediante CSG. El asa se creó utilizando un Torus ligeramente escalado.
  - *Color:* Para el botiquín se utilizaron colores básicos. La estructura principal es roja, la cruz es blanca y el asa es negra.

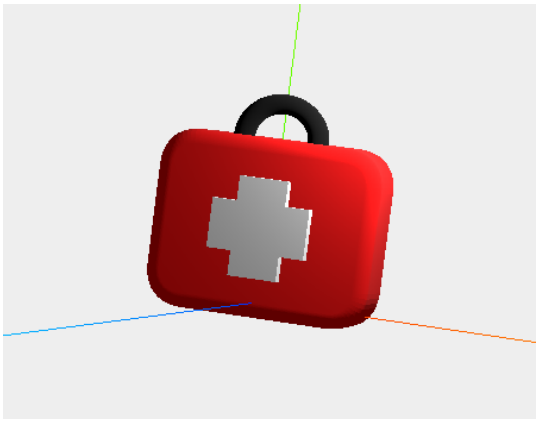

Figura 3: Botiquín

- **Estrella**
  - *Descripción:* Para crear una estrella más realista, obtuvimos los puntos de sus vértices utilizando la herramienta GeoGebra. Empezamos dibujando un polígono regular y luego uniendo los vértices para formar la estrella. Con estos puntos, creamos una forma (`THREE.Shape`) en Three.js y la extruimos para darle volumen.

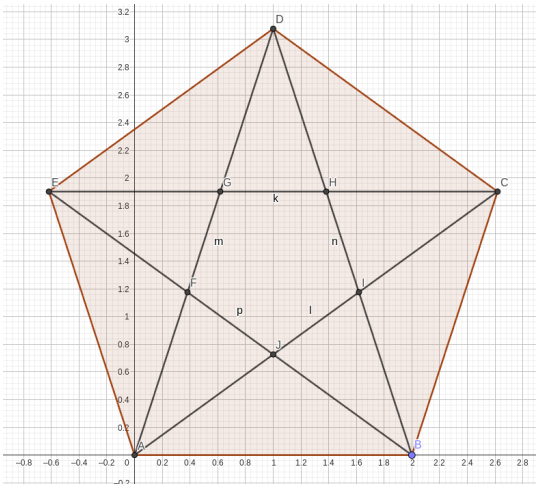


Figura 4: Obtención de los puntos de la estrella

- *Color:* La estrella tiene un único color amarillo y una luz dentro del juego que comentaremos más adelante.

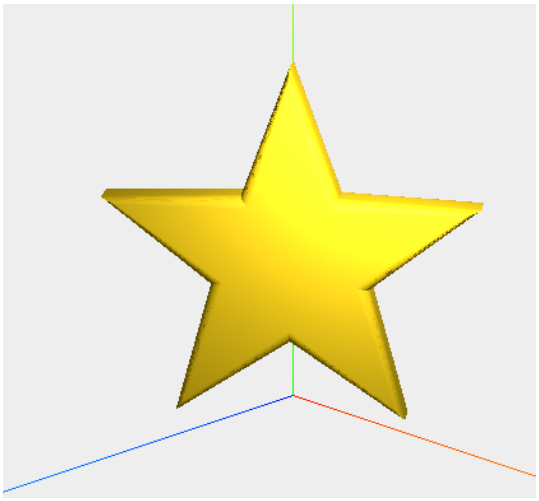

Figura 5: Estrella

- **Bandera**
  - *Descripción*: La bandera se modeló utilizando una forma (`THREE.Shape`) con curvas cuadráticas para crear un diseño ondulado y realista. La forma se extruyó para darle volumen, y se añadió un cilindro para representar el mástil.
  - *Textura:* La textura de la bandera se carga desde una imagen de una bandera.

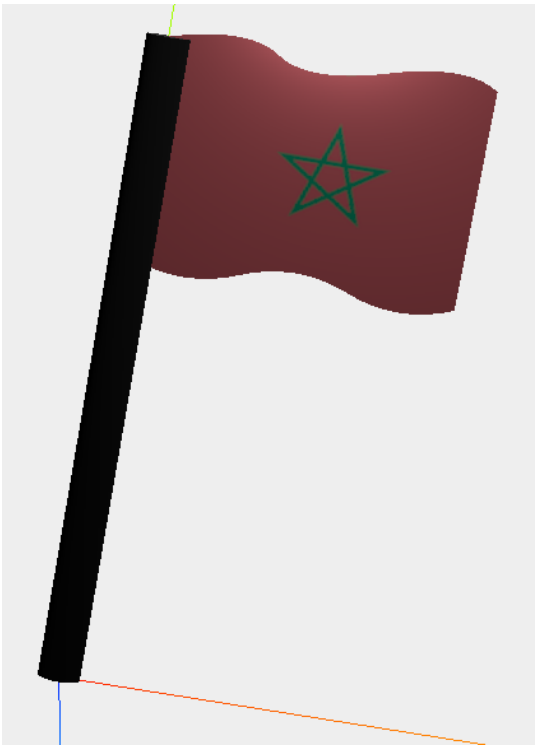

Figura 6: Bandera

- **Bomba**
  - *Descripción:* La bomba está compuesta por una esfera principal que representa el cuerpo de la bomba, un cilindro pequeño para la cabeza y otro cilindro aún más pequeño para la mecha. Estos componentes se unen en el objeto final.
  - *Color:* La bomba está formada por colores básicos, la esfera principal de color negro, el cilindro para la cabeza blanco y la mecha marrón. Además, se añade una luz roja intermitente para simular la activación de la bomba.

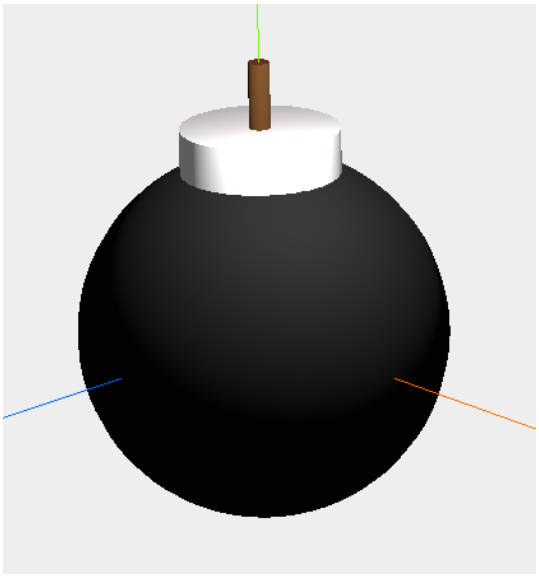

Figura 7: Bomba

- **Muro**
  - *Descripción:* El muro es el objeto más simple que tenemos, es una caja con las medidas ajustadas para darle forma de muro.
  - *Textura:* Para la textura del muro, hemos usado 3 imágenes distinas, una para el color, otra para la rugosidad y otra para la normal.

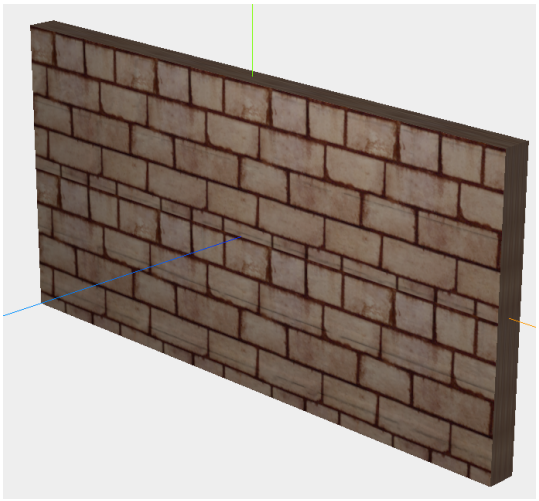

Figura 8: Muro

- **Luna**
  - *Descripción:* La luna es el único objeto del proyecto que hemos importado desde un modelo 3D existente en internet.
  - *Textura:* Para la textura de la luna, hemos utilizado una textura simple que resalta los detalles del modelo, proporcionando una apariencia más realista. La textura se puede encontrar en el enlace: `https://free3d.com/3d-model/man-in-the-moon-v1--337906.html`

Figura 9: Luna

- **Misil**
  - *Descripción:* El misil está compuesto por un cuerpo principal generado mediante revolución a partir de un (`THREE.Shape`) con forma de curva. Las patas del misil se crean extruyendo otro (`THREE.Shape`) y luego se clonan y posicionan adecuadamente alrededor del cuerpo.
  - *Color:* El misil utiliza colores básicos: el cuerpo es blanco, las patas son negras y cuenta con una luz roja parpadeante.

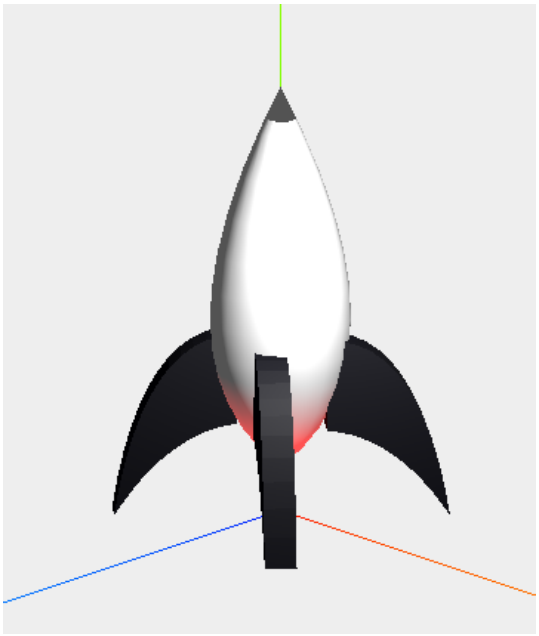

Figura 10: Misil

- **Dron**
  - *Descripción:* Para el cuerpo del dron, hemos creado una esfera y la hemos achatado. Las hélices se componen de tres cilindros: el cilindro principal con el radio inferior más pequeño y escalado para darle la forma característica, el soporte que es un cilindro más pequeño y normal, y el último es un cilindro achatado que forma la hélice.
  - *Textura:* Para el dron, al igual que en el tubo, decidimos aplicar una textura con relieve con forma metálica, obtenida del siguiente enlace:
    `https://3dtextures.me/2024/05/22/metal-pierced-001/`

# 3. Algoritmos Usados Mas Importantes

- *Algoritmo de deteccion de Colisiones:*
  Para detectar las colisiones lo primero que se hace es crear los rayos en la posicion inicial del tanque con la direccion en la que mira el

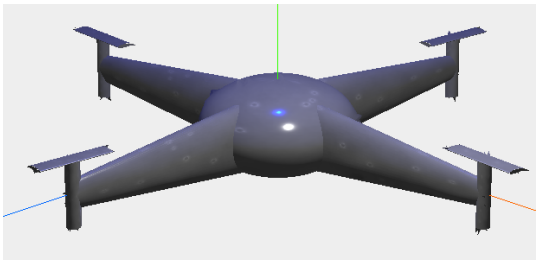

Figura 11: Dron

tanque. Este paso se realiza en el constrcutor de la Escena en el metodo CreateRayos.

En este metodo se posicionan y crean los rayos unicamente, estos rayos se guardan en un array de Rayos que utilizaremos posteiornmente, pero esto no basta para poder detectar las colisiones. Tenemos que actualizar la posicion de los rayos conforme el tanque se vaya moviendo, esto se hara el el metodo updateRayos que se ejecutará continuamente en el update de le escena.

En este metodo de updateRayos lo que hacemos recorrer todos los rayos y actualizar la posicion actual del rayo a la misma que la del tanque y guardar en un array los objetos con lo que ha colisionado si los hay. En caso de que no haya colisonado con ningun objeto seguira su curso, pero si ha colisonado con un objeto se recorrera todos los objetos con los que se han colisionado y se vera que tipo de objeto es, si es un objeto que nos beneficia, perjudica o si es indiferente. Para de este modo poder actuar en consecuencia y restar o sumar puntuacion o vida.

- *Algoritmo de Picking:*
  - *Definición de variables:*
    - *pickableObjects:* Vector con los objetos a los que vamos a poder hacerle picking en nuestro juego.
    - *raton:* Vector donde guardamos las coordenadas del ratón (x,y).

- *raycasterRaton:* Objeto de tipo (`THREE.Shape`) para lanzar rayos desde la cámara hacia los objetos en la escena.

- *Función de selección 'onDocumentMouseDown':* Esta función se dispara cuando el usuario hace click en la ventana del juego. Obtenemos las coordenadas del ratón en la ventana y lanzamos un rayo desde la cámara hacia los objetos en la escena, usando las coordenadas normalizadas del ratón.

  Luego, utilizamos el método intersectObjects del raycaster para determinar qué objetos están siendo intersectados por el rayo y lo guardamos en la variable pickedObjects.

  Por último, comprobamos si la longitud del vector pickedObjects es mayor que 0, lo que significa que hemos colisionado con algún objeto. En ese caso, hacemos invisible al objeto seleccionado y aumentamos la puntuación en 30.

- *Algoritmo de Generacion de objetos:*
  Este algortimo genera un numero determinado de objetos en posiciones y angulos aleatorios en cada vuelta. Esto se hace para dinamizar el juego y que sea más divertido.

  Se ha programado de la siguiente manera: En cada vuelta se eliminan todos los objetos de la escena y se van creanado poco a poco los objetos nuevos, dependiendo del numero de objetos que se haya especificado en el metodo.

  Por ejemplo si hemos puesto que queremos crear 3 Drones mediante un bucle-for creamos esos tres drones. O si queremos crear 10 botiquines creamos 10 botiquines mediante un for, a estos objetos se les pasan unas coordenadas por parametro generadas de forma aletoria para que aparezacen en un sitio diferente en cada vuelta que da el tanque.

  Tambien se añadiran estos objetos al array que controla que objetos pueden colisionar o a cuales se les puede hacer picking.

Solo el tanque sabe cuando se ha completado una vuelta, entonces el tanque tendra una instacia de mi escena y cada vez que se de una vuelta llamara a este metodo para generar los nuevos objetos.

- *Algoritmo de Movimiento del Personaje Y Cambio de Cámara:*

  - Función de movimiento 'onKeyDown': Esta función se activa cuando el usuario presiona una tecla en el teclado.
    Se obtiene la tecla presionada mediante el atributo key del evento.
    Se utiliza una estructura de control switch para determinar qué tecla se ha presionado.
    - 'ArrowLeft': Si la tecla presionada es la flecha izquierda, se llama al método girarIzda() del tanque para que gire hacia la izquierda.
    - 'ArrowRight': Si la tecla presionada es la flecha derecha, se llama al método girarDerecha() del tanque para que gire hacia la derecha.
    - 'Espacio': Si la tecla presionada es la barra espaciadora, se verifica si la cámara principal está activa o no. Si no lo está, se cambia la cámara principal a la cámara general (camera). Si ya está activa, se cambia la cámara principal a la cámara del personaje del tanque (getCameraPersonaje()).
    - Default: En caso de que se presione otra tecla que no sea ninguna de las anteriores, no se realiza ninguna acción.

- *Algoritmo de Animación de Recorrido:* Lo primero es crear una animación de la librería TWEEN.js.
  La animación se repite indefinidamente y actualiza la posicion del tanque.
  Además, cada vez que el objeto en movimiento completa una vuelta completa alrededor del path (es decir, cuando el parámetro t del origen es un múltiplo entero de 1), se incrementa un contador de vueltas, se reduce el tiempo de cada frame en un 10 %, y se vuelve a establecer el tiempo mínimo en 20000 ms. También se vuelven a añadir a la escena todos los objetos del mapa. Por último, se inicia la animación con el método start().

La animación para cuando el juagdor pierde la partida.

# 4. Manual De Usuario

A continuación, se describen las características y utilidades de cada objeto en el juego, así como su impacto en el personaje principal. El tanque se moverá continuamente hacia adelante, y los jugadores pueden usar las teclas de flecha para moverlo hacia la izquierda o la derecha. El juego ofrece dos vistas de cámara distintas, y los jugadores pueden cambiar entre ellas presionando la tecla .Espacio".

- **Personaje principal**
  - *Tanque:* El tanque es el personaje principal que recorre el tubo, interactuando con diversos objetos que pueden beneficiarlo o perjudicarlo. Estará en continuo movimiento
- **Objetos Puntuadores**
  - *Bandera:* La bandera es la forma más básica de obtener puntos. Al recoger una bandera, se suman 10 puntos a la puntuación final. En cada vuelta habrán 20 banderas para que el usario pueda puntuar hasta 200 punto si obtiene todas.
  - *Misil:* Los misiles son objetos voladores que recorren el tubo en dirección contraria al tanque. Al disparar y destruir un misil, se obtienen 30 puntos. En cada vuelta apareceran 3 misiles distribuidos en posiciones aleatorias. Es dificil acertar el disparo a un misil por eso tiene una gran recompensa.
  - *Dron:* El dron es un objeto volador que se mueve alrededor del tubo a mayor velocidad que los misiles, lo que dificulta dispararle. Destruir un dron otorga 30 puntos ya que tambien es muy dificil dispararle en movimiento.Apareceran 3 drones por vuelta para tener más posibilidades de darle.
- **Objetos negativos**

- *Bomba:* La bomba resta 10 puntos de vida al tanque al entrar en contacto con él. Apareceran 12 bombas en cada vuelta de forma aleatoria, para dificultar el juego un poco.
- *Muro:* El muro aparece aleatoriamente sobre el tubo y resta 30 puntos de vida al tanque si colisiona con él.Dado que quita mucha vida y abarca mucho terreno solo habran 3 muros por vuelta.A veces puede ser dificil esquivarlo y te llevas una gran penalización.

- **Objetos Positivos**

  - *Botiquín:* El botiquín restaura la vida del tanque, permitiendo que el juego continúe si ha perdido vida debido a la interacción con objetos negativos. Aumenta 10 de vida y es un objeto importante para poder llegar lo más lejos posible. Habrán 6 botiquines por vuelta.
  - *Estrella:* Al obtener una estrella, el tanque se vuelve inmune a los objetos negativos durante 10 segundos,obtiene el doble de puntos por cada bandera recogida y otorga 10 puntos al jugador. Ya que es un objeto con muchos beneficios solo aparece 1 vez por vuelta. Puede ser clave para llegar lejos en la partida.
  - *Luna:* La luna es un elemento decorativo que no afecta al juego. Se puede disparar para que desaparezca, pero esto no altera la puntuación ni el estado del juego.

# 5. Diagrama De Clase

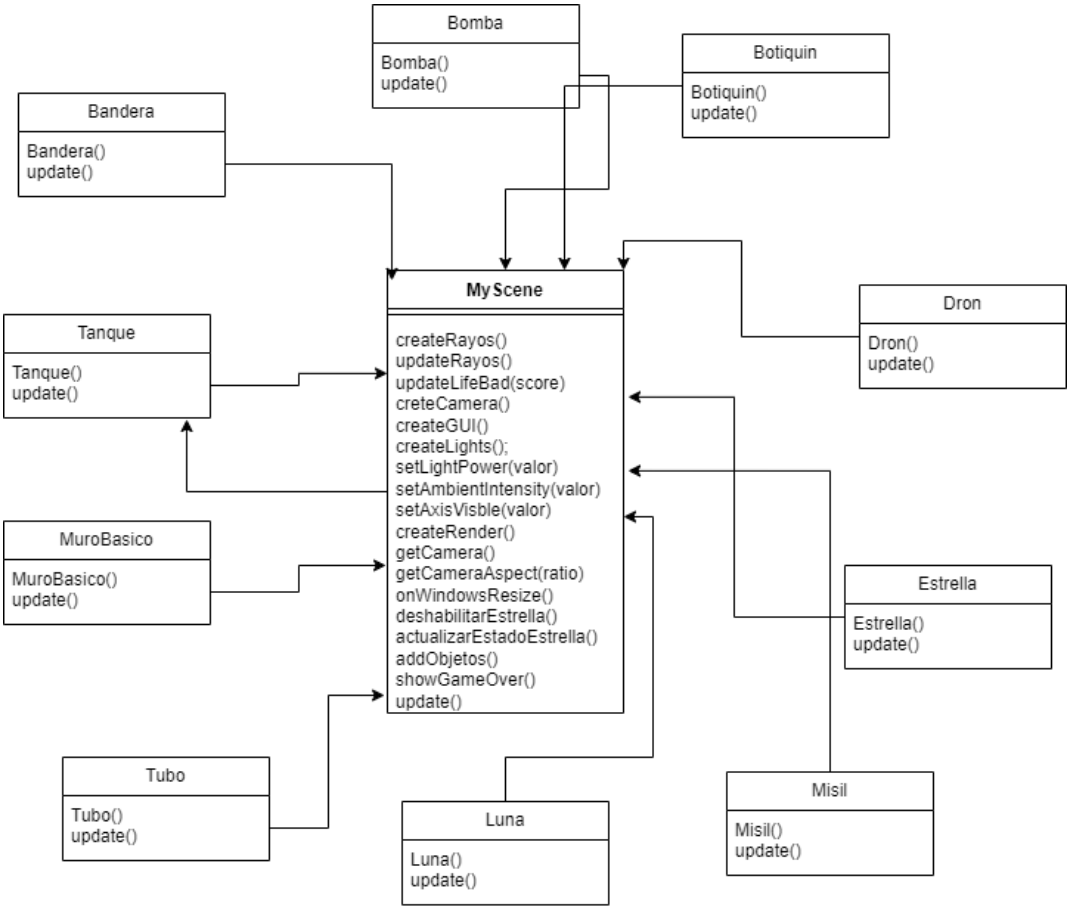


Figura 12: Diagrama de clases