Marta Rincón Otero

Anne Serrano Andrades

***ANOTACIÓN$_1$:** Debido a la generación aleatoria de los objetos entre otras razones, la práctica tarda mucho en cargar. Acceso al juego: practica2_SG/src/. Acceso a los objetos para su visualización individual: practica2_SG/Objetos/<nombre_del_objeto>*
***ANOTACIÓN$_2$:** Acceso al código de la práctica https://github.com/marta021/practica2_SG.git*

Descripción

RESUMEN DEL JUEGO:

- **OBJETIVO:** Conseguir el máximo número de puntos.
- **SISTEMA DE PUNTUACIÓN:** Los puntos se obtienen disparando a los objetos voladores.

PERSONAJE PRINCIPAL: El personaje principal tendrá la forma de un coche que será un modelo cargado desde disco.

ELEMENTOS DEL SUELO:

Se caracterizan por objetos “buenos” y “malos” según la consecuencia que tengan.

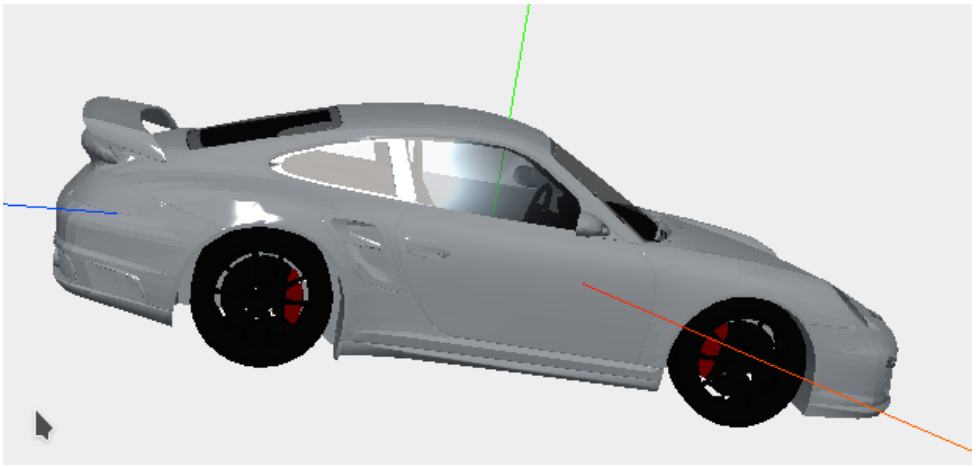
- Seta: aumento de velocidad de un 20%. Se trata de un objeto por revolución.
- Pincho: reducción de velocidad de un 10%, objeto hecho mediante csg.
- Rayo: reducción de velocidad de un 10%, pérdida de 5 puntos y luz oscura durante 5 segundos

ELEMENTOS VOLADORES:

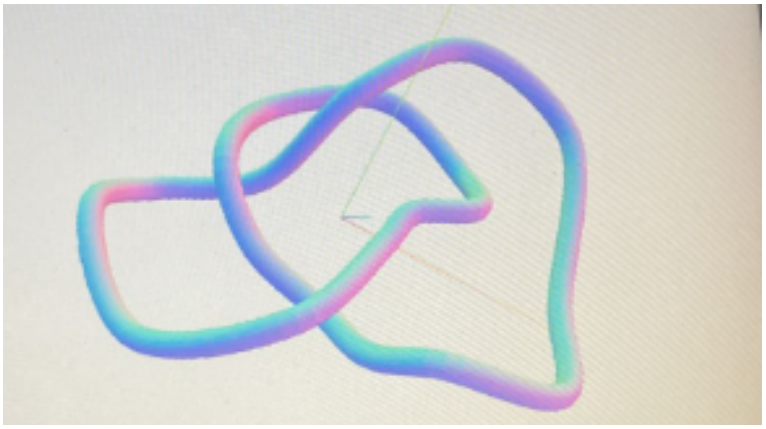
- Estrella: Proporciona velocidad (+15%) y 10 puntos..
- Nube: No tiene ningún beneficio aparte y suma 5 puntos.
- Fantasma: No tiene ningún tipo de beneficio aparte de sumar 10 puntos.

BOCETOS/ MODELOS DE LOS OBJETOS:

PERSONAJE:

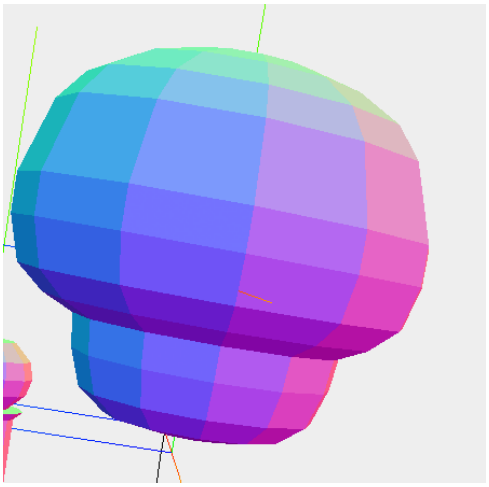


CIRCUITO:



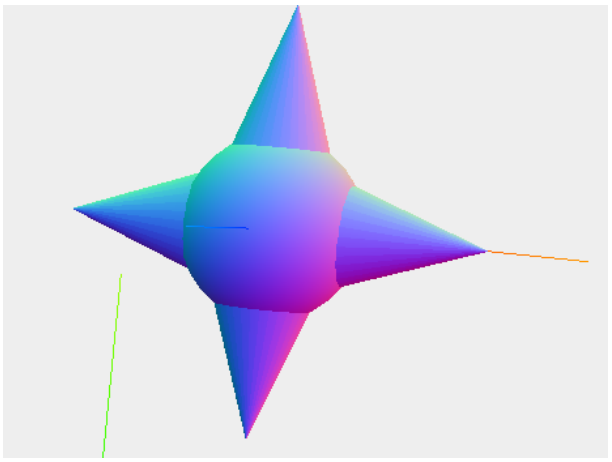
(sin aplicación de texturas/materiales)

SETA:



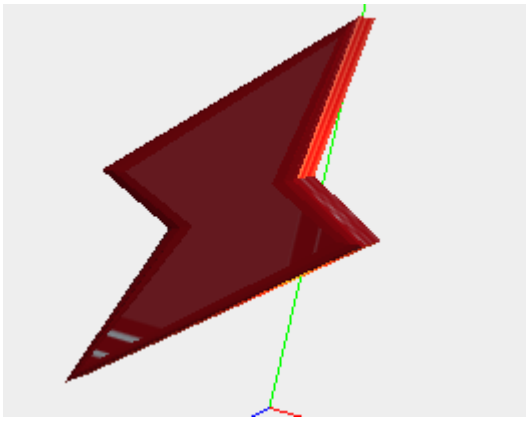
(sin aplicación de texturas/materiales)

PINCHO:



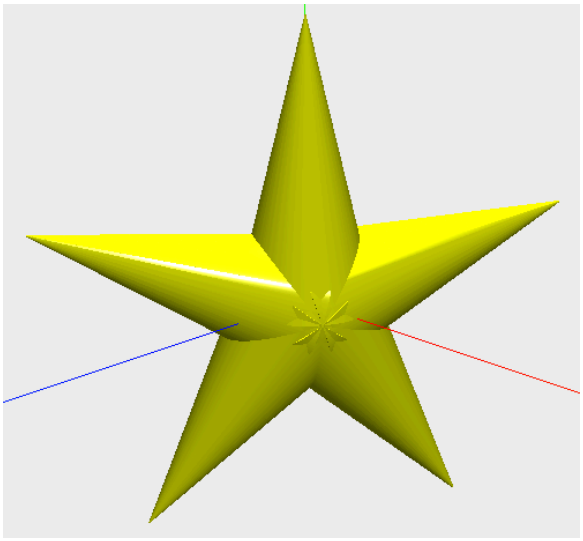
(sin aplicación de texturas/materiales)

RAYO:



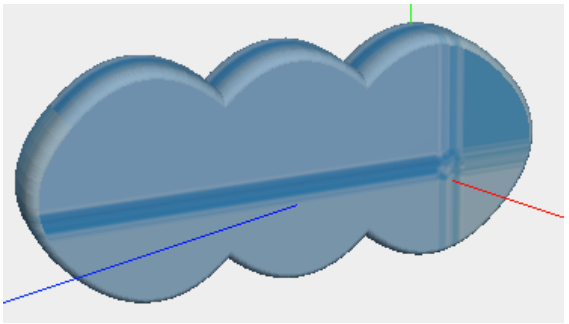
(con aplicación de texturas/materiales)

ESTRELLA:



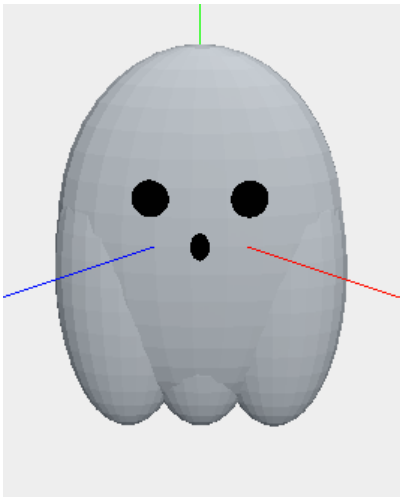
(con aplicación de texturas/materiales)

NUBE:



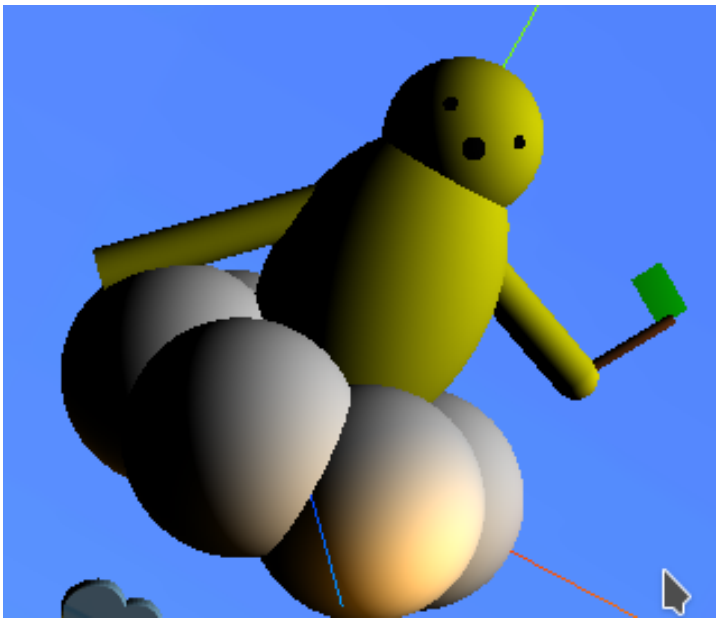
(con aplicación de texturas/materiales)

FANTASMA:



(con aplicación de texturas/materiales)

OBJETO ARTICULADO: personaje que mueve uno de sus brazos constantemente hacia arriba o hacia abajo (brazo derecho) . En dicho brazo tendrá una bandera que también se moverá rotando arriba y abajo.



ANIMACIÓN

- *Coche:*

- A cada vuelta que complete se aumenta su velocidad un 10%.
- Además la velocidad se verá afectada según los objetos con los que colisione / dispare, según como se ha comentado en el apartado de arriba en la descripción de cada objeto.
- Su movimiento hacia delante es constante, sin embargo; puede moverse hacia los laterales (alrededor del tubo) con el uso de las teclas “a” (rotación hacia la izquierda) y “d” (rotación hacia la derecha).

- *Objeto articulado:*
- *Para cambiar el punto de rotación de los objetos solo necesito agregarlo a un punto de pivote, la animación se actualiza en la escena, creando una oscilación del brazo.*

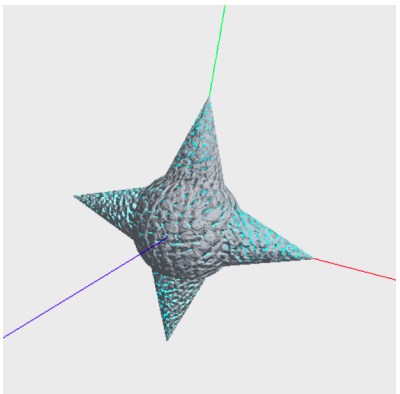
- *Objetos voladores:*
- *El movimiento de los objetos voladores se define por una rotación contínua en el eje y, la cual actualizamos en la escena.*

INTERACCIÓN

- *Vista en 3ª persona:*
- Como se ha comentado anteriormente, el personaje puede desplazarse usando el teclado hacia la derecha e izquierda.
- Al hacer clic en los objetos voladores estos proporcionarán las ventajas o desventajas comentadas anteriormente (suma o pérdida de puntos o velocidad, entre otros).
- *Vista en general:*
- Tras pulsar la tecla “espacio” se alterna entre esta vista y la anterior mencionada, donde se permite alterar el punto de vista arrastrando el ratón. Además, con la rueda del ratón se puede hacer zoom.

MATERIALES

- *Materiales basados en color:*
  - *Estrella*
  - *Fantasma*
  - *Seta*
- *Materiales basados en textura para el canal de color:*
  - *Nube*
  - *Rayo*
- *Materiales basados en textura para el canal de relieve:*
  - *Pincho*



LUCES

- *Luces de colores:*
  Podemos observar 4 tipos de luces puntuales (PointLight) de diferentes colores:
- seta (roja).
- rayo (naranja).
- pincho (azulado).
- estrella (blanco).

- *Luz que cambia:*
  Como consecuencia de la colisión con un objeto de tipo *rayo* la luz puntual que se encuentra en la escena se ve modificada. Su intensidad pasará a ser 0.0 durante 5 segundos, pasado este tiempo su intensidad volverá a verse modificada a 0.8.

CÁMARAS

- *Cámara en tercera persona:*

Para crearla, se crea un objeto con el constructor PerspectiveCamera con sus respectivos parámetros, la cual colocamos con una distancia positiva en el eje y, y a una distancia negativa del eje z para que cuando se la añadamos al objeto coche esté justo detrás de él y un poco por encima suya. Al añadirla al objeto coche está cámara se quedará colgando de él toda la animación

- *Cámara en tercera persona:*

Esta cámara es la cámara clásica que tenemos creada en otras escenas que apunta al centro de coordenadas. Desde donde podemos ver todos los elementos.

Diseño.

Diagrama de clases

Anexo en la carpeta documentación, png llamado “diagrama_clases.png”.

Modelo jerárquico

Anexo en la carpeta documentación, pdf llamado “Modelo jerárquico.pdf”

Descripción algoritmos

COLISIÓN

A continuación, expresaremos en pseudocódigo el algoritmo de colisiones para su posterior explicación:

```
// Pseudocódigo para manejar las colisiones en MyScene.js
funcion testColision(posicion, direccion) {
    direccion = normalizar(direccion)
    raycaster = establecerRaycaster(posicion, direccion)
    colisiones = raycaster.intersectarObjetos(obstaculos.hijos,
true) // true para incluir hijos de hijos

    // Si hay colisiones y la distancia de la primera colisión es
menor a 0.3
    si (longitud(colisiones) > 0 y colisiones[0].distancia < 0.3)
{
        objetoColisionado = colisiones[0].objeto.padre.nombre
        segun (objetoColisionado) {
            caso 'pincho':
                // Eliminar objeto colisionado

colisiones[0].objeto.padre.eliminar(colisiones[0].objeto)
                coche.colisionPincho = verdadero
                romper

            caso 'seta':
                // Eliminar objeto colisionado

colisiones[0].objeto.padre.eliminar(colisiones[0].objeto)
                coche.colisionSeta = verdadero
                romper

            caso 'rayo':
                // Eliminar objeto colisionado

colisiones[0].objeto.padre.eliminar(colisiones[0].objeto)
                coche.colisionRayo = verdadero
                colisionDesventaja = verdadero
                puntuacion -= 5
```

```
                romper
        }
    }
}

// Pseudocódigo para manejar las consecuencias de las colisiones
en coche.js
funcion update() {
    si (coche.colisionPincho) {
        coche.setVelocidad(0.9)
        coche.colisionPincho = falso
        imprimir("Colisión con pincho: VELOCIDAD REDUCIDA")
    } sino si (coche.colisionSeta) {
        coche.setVelocidad(1.2)
        coche.colisionSeta = falso
        imprimir("Colisión con seta: VELOCIDAD AUMENTADA")
    } sino si (coche.colisionRayo) {
        coche.setVelocidad(0.9)
        coche.colisionRayo = falso
        imprimir("Colisión con rayo: VELOCIDAD REDUCIDA")
    } sino si (coche.pickEstrella) {
        coche.setVelocidad(1.15)
        coche.pickEstrella = falso
        imprimir("Pick con estrella: VELOCIDAD AUMENTADA")
    }
}


// Pseudocódigo para la parte de las consecuencias de las
colisiones del método update de MyScene.js
funcion update () {
    si (coche.colisionDesventaja) {
        reloj.iniciar()
        coche.setLightIntensity(0.0)
        imprimir("ATENCION: consecuencia de colision con rayo
ACTIVADA (se va la luz)")
        coche.colisionDesventaja = falso
        coche.consecuenciaDesventaja = verdadero
    }

    si (coche.consecuenciaDesventaja y
reloj.obtenerTiempoTranscurrido() > 5) {
        coche.setLightIntensity(0.8)
        imprimir("ATENCION: consecuencia de colision con rayo
DESACTIVADA (vuelve la luz)")
        reloj.detener()
        coche.consecuenciaDesventaja = falso
    }
}
```

Para realizar las colisiones lanzaremos un rayo desde el centro del coche y almacenaremos todos los objetos con los que ha impactado en el vector `colisiones`. Tenemos en cuenta

el rayo según `obstaculos.hijos` ya que estos “obstaculos” que se mencionan son los dispuestos en el tubo. Tomamos el valor de longitud como “0.3” para que la colisión sea dada a partir del morro del coche y que no se tenga que esperar a que el objeto colisione con el centro de este para ser considerada una colisión.

Una vez se ha colisionado se procede a establecer las consecuencias pertinentes según el caso, es por ello que establecemos variables booleanas acorde además de eliminar el objeto del tubo. En el caso de la colisión con el objeto “rayo” también se modifica la puntuación, además de alterar la luz ambiente, esto se ve en el `update` de MyScene.js. En este método, se usa la función `setLightIntensity` para ello. Para saber cuándo devolver la luz de nuevo se usa un reloj interno que es activado en el momento de la colisión y que se para una vez transcurridos los 5 segundos de la consecuencia.

Por último mencionar como la velocidad se ve alterada según el objeto colisionado según la variable booleana activada. En `coche.js` se hace uso de la función auxiliar `setVelocidad` donde simplemente se le asocia a la variable `Velocidad` el nuevo valor correspondiente según el factor que se le ha pasado por parámetro.

DISPARO

A continuación, expresaremos en pseudocódigo el algoritmo del disparopara su posterior explicación:

```
// Pseudocódigo para manejar el disparo (picking)en MyScene.js
onMouseDown(event):

  // Obtener la posición del clic del mouse en coordenadas
normalizadas
  mouse.x = (event.clientX / window.innerWidth) * 2 - 1
  mouse.y =  -(event.clientY / window.innerHeight) * 2 + 1

 // Actualizar el raycaster con la posición del clic del mouse y
la cámara actual
  raycaster.setFromCamera(mouse, getCamera())

// Realizar la intersección del rayo con los objetos voladores
intersects = raycaster.intersectObjects(voladores.children, true)

  // Verificar si se han encontrado intersecciones
  if intersects.length > 0:
   // Seleccionar el primer objeto intersectado (el más cercano
al    clic del mouse)
    objetoSeleccionado = intersects[0].object

  si (objetoSeleccionado==estrella) {
        puntuacion+=10
        coche.pickEstrella = true
        borrarObjetoSeleccionado

        imprimir("Pick Estrella")
    } sino si (objetoSeleccionado==nube){
```

```
        puntuacion+=5
        borrarObjetoSeleccionado
        imprimir("Pick Nube")
    } sino si (objetoSeleccionado==fantasma){
        puntuacion+=15
        borrarObjetoSeleccionado
        imprimir("Pick Fantasma")

    }
```

Para poder realizar la selección de objetos cuando el usuario hace click con el raton, se necesita crear un evento de click ‘onMouseDown’, este evento se activa cuando se hace click sobre la escena. Cuando hacemos click se obtiene la posición x e y del ratón, dicha posición se usa para actualizar el rayo ‘raycaster’.
Se realiza la intersección del rayo con los objetos que hay en la escena, para obtener qué objetos están bajo el puntero del ratón.

Si el rayo intersecta con alguno de los objetos voladores, se selecciona el primer objeto encontrado, y se identifica el nombre del objeto (estrella, fantasma o nube) el cual se obtiene según el nombre de su padre en la jerarquía. Dependiendo del tipo de objeto seleccionado (estrella,fantasma o nube) se hacen unas acciones específicas:

- Estrella: si el objeto es una estrella se aumenta la puntuación, se llama a pickEstrella que aumentará la velocidad del coche y se elimina la estrella de la escena.
- Nube: si el objeto es una nube se aumentará la puntuación y se borrará la nube de la escena.
- Fantasma: si el objeto es un fantasma se aumentará la puntuación y se borrará el fantasma de la escena.

Para la actualización de la puntuación se va aumentando el valor this.puntuacion según la interacción con los objetos y se muestra en el código html.

Para eliminar los objetos de la escena usamos el padre del objeto seleccionado eliminandolo de la jerarquía de objetos de la escena.

Manual de usuario

El objetivo del juego es conseguir el máximo número de puntos. Para ello, el coche ha de desplazarse por el tubo colisionando con los objetos y disparando a los objetos voladores pertinentes.
Objetos con los que debería colisionar:

- **Seta** para aumento de la velocidad.

Objetos con los que debería evitar colisionar:

- **Pincho** para evitar perder velocidad.
- **Rayo** para evitar que se vaya la luz y perder puntos.

Para la obtención de puntos (objetivo principal del juego), el personaje debería disparar a **todos** los objetos, ya que todos ellos proporcionan puntos positivos. Además, si se desease velocidad el objeto **estrella** la aumenta, con lo cual sería el mejor objeto a pillar para obtener puntos y ganar rapidez al mismo tiempo.

Los objetos que dan puntuación son:

- **Estrella**: 10 puntos. Esta además proporciona velocidad.
- **Nube**: 5 puntos.
- **Fantasma**: 15 puntos.

Bibliografía

- El personaje principal lo he descargado de https://sketchfab.com/3d-models/porsche-911-gt2-b30dffc4d32c49bfa66db4aa6972bba5
- Hemos actualizado la biblioteca de Tween de la vista en clase con https://github.com/tweenjs/tween.js/blob/main/src/Tween.ts