DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Asteroids 3D

Final

Fundamentos de la Computación Gráfica

Primer Cuatrimestre de 2021

Integrante	LU	Correo electrónico
Daniel Lopera	948/19	lopera.dani@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

El objetivo de este trabajo práctico consistía en implementar un proyecto final para la materia **Fundamentos de la Computación de Gráfica**. Entre las propuestas se eligió el proyecto de **Asteroirds 3D** en el que consistía en escribir un juego similar al Asteroids pero en 3D. En este informe se habla sobre las decisiones tomadas para lograr el proyecto y explicación del código. He de acotar que el proyecto fue realizado únicamente en Javascript con el framework Three.js y un poco de Html y CSS para la interfaz. Es importante aclarar que por falta de conocimiento en el lenguaje, se tuvo que estudiar y aprender sobre la marcha, por lo que hay decisiones que no tienen buen performance o ideas que en ciertas clases son mejoradas y en otras no.

Para acceder al proyecto, se puede descargar desde el siguiente repositorio junto a las instrucciones de instalación y los controles del juego.

Repositorio: https://github.com/daniellopera15/Asteroids3D

2. Lógica del juego

Nota: El juego está implementado con la idea de mantener una armonía POO.

Al iniciar la página, se instancia la clase **Game** (como controlador del juego) donde comenzará a implementar lo necesario para que el juego pueda funcionar en su constructor, todo esto ocurrirá sin que el juego inicie, pues se espera a que el jugador le de al botón de **Play**.

Se crea una cámara y se coloca en cierta posición y se rota. (Por error al momento de implementar la Nave, quedó en una dirección errónea y al tener varios problemas al implementar el movimiento, se roto la cámara para evitar más problemas y solucionarlo)

Se cargan los sonidos primero para que estén cargados para cuando se inicie el juego. Se toma un clock y un delta para poder limitar los fps, con la intención de que en máquinas más poderosas el juego pueda ser disfrutado. Se crea la escena donde se agregan todos los elementos que aparecen en el juego, junto a él se carga una imagen en el background para que sea el fondo de estrellas en el juego. Además de esto se implementa una función que se ejecutará para adaptar el juego por si se cambia el tamaño de la página web.

A continuación se ejecuta la función **load**, esta es la encargada de crear los elementos necesarios que estarán listos, como el renderer, la iluminación, los bordes del juego, la nave y las variables que se usan para controlar las balas, asteroides y las explosiones.

2.1. Inicio de juego

El jugador solo observará a la nave en el medio de la pantalla y al hacer click, comienza el juego. En la función **startGame** se desaparecen los botones, se limpian los asteroides (pues también servirá cuando se reinicie el juego), se inicializan las vidas y el puntaje del jugador, comienza la música de fondo y se deja un limite de asteroides creados por partida, de esta manera no se produce un descontrol de exceso de asteroides en pantalla.

Se ejecuta **startAsteroids**, función que se encargará de crear asteroides cada 2,5 seg. La creación de los asteroides será aleatoria, tanto su tamaño, como de cual borde incia y la dirección en la que saldrá el asteroide.

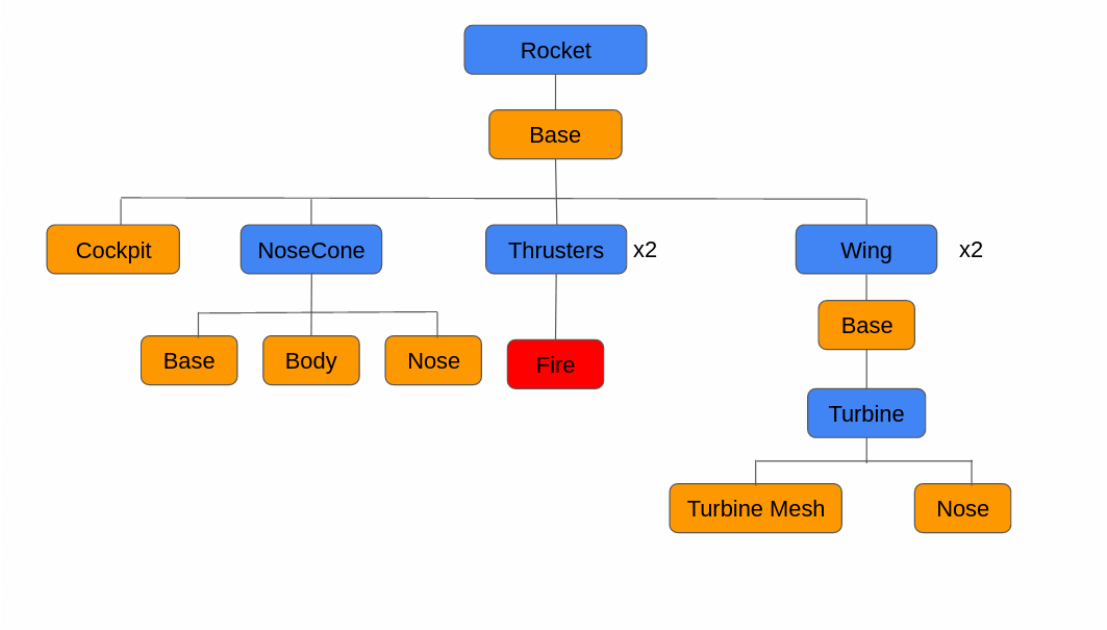
La función **render** será la encargada de hacer funcionar el juego. Ejecutará el update de cada objeto que lo requiera (Nave, Asteroide, Bordes, Disparos y Explosiones) al igual que con las colisiones. En cuanto a los Asteroides, Disparos y Explosiones, son elementos que aparecen, y desaparecen manejan una lógica parecida, pues al ser varios, sus instancias son guardadas en arrays y cada uno tiene un flag que indica si aún están existiendo en la pantalla o ya fueron eliminados, de está manera al ser eliminados se limpian de los arrays antes de llamar los updates.

Al jugador perder las 3 vidas, se ejecuta la función **gameOver** donde se ejecuta un sonido para el gameOver, aparece una frase de que se finalizó el juego y un botón para reiniciar y volver a jugar.

3. Modelos

3.1. Rocket

La nave fue inspirada en la ”Gummi Ship”del videojuego **Kingdom Hearts** y fue implementada en su totalidad con figuras proporcionadas por Three.js se implementaron las figuras y se rotaron y agregaron en cierto orden para darle forma a la nave. Quedando de la siguiente forma:



Para la nave se crea una esfera que está alrededor de ella para poder crear las colisiones con los Asteroides, y se crean listener para atender los eventos por teclado, tanto para el momento en que se presiona una tecla y se suelta.

Además inicializan los valores de la nave, como las vidas, la nave en reposo, inicia en el centro de la pantalla y se agrega a la escena.

Los eventos por el teclado se ejecutarán solo si el juego se está ejecutando y si hay vidas disponibles. Los disparos solo estarán disponibles uno por cada vez que se presione una tecla, es decir que al presionar la barra espaciadora se instanciará una bala y no se creará otra hasta que se suelta la tecla y se vuelva a presionar. Para los movimientos solo se podrá rotar hacia los lados y acelerar hacia adelante y en este evento se activa el sonido de la nave y aparecerán los fuegos, los cuales serán desactivados al soltar la tecla.

Para el movimiento de la nave, se incia con los valores en 0 por tener la nave en reposo y al presionar la tecla correspondiente se irá acelerando mientras se tenga presionado hasta alcanzar un limite. De igual manera ocurrirá al soltar la tecla, este irá frenando de a poco generando un efecto de inercia, tanto para el movimiento de rotación como el de aceleración.

Al actualizar la nave, se actualizan los fuegos si estos se encuentran activados, se actualizan los valores para generar el movimiento.

Cuando la nave es destrida, se desactivarán los eventos de movimientos, se reproducen los sonidos correspondientes, se decrementa la vida del jugador y se instanciará una explosión de color blanco. Luego de 2 seg, la nave reaparecerá en el centro y producirá un efecto intermitente por 3 seg y durante ese tiempo la nave no podrá ser destruida. Una vez que haya perdido todas las vidas, está le indicará a la clase Game que ejecute el gameOver.

3.2. Bullet

Al ser instanciada una bala, se copia la dirección de la nave y se rota para poder quedar en dirección de la nave y al ser disparada pueda producir el efecto de un disparo. La forma geometrica de la bala es un cilindo y esta rodeada de un box3 para poder producir la colisión con los Asteroides. Las Balas al ser disparadas avanzarán en una sola dirección y tienen una distancia que recorrer. Ella avanzará hasta que llegué a su limite o se encuentre con un Asteroide en el camino, de ambas formas la bala será removida. El movimiento de la bala siempre será el mismo, con una velocidad constante.

3.3. Asteroid

La implementación gráfica de los Asteroides fue extraída de un código que compartió el profesor y fue adaptado para que funcionará con la lógica del juego y se extrajo solo lo necesario. Todos tienen una esfera para poder producir las colisiones con las Balas y la Nave.

Existen 3 tipos de Asteroides, Tipo A, B o C. Estos definen las caracteristicas de cada una, empezando por A siendo el más grande y el C el más pequeño. Cada tipo definirá el valor del asteroide, la cantidad de vidas que tendrá cada uno, la velocidad, su iluminación y sus dimensiones que será aleatorio acorde a un rango definido para el tipo.

Al ser eliminado un Asteroide, este producirá una explosión de color rojo y si es de tipo A, se producirán 3 Asteroides de tipo B, al eliminar uno de B se crean 2 de tipo A en su lugar, y al eliminar de tipo C desaparecen. Además le asignarán al jugador el puntaje correspondiente.

Para poder controlar la cantidad de Asteroides por pantalla, se le asigna un valor al asteroide acorde a su tipo y va relacionado con los Asteroides que tiene anidados, siendo las de tipo C con el valor de 1, B con 2 y por último C con 6.

3.4. Edge

Se instancian cuatros bordes, uno para cada lado de la pantalla. Su objetivoconsiste en estar pendiente por si tiene algún elemento cerca, sea la Nave, Bala o Asteroide, y al conseguirlo hace que el elemento aparezca del otro lado de la pantalla. Visualmente no son visibles pero esan ubicados al borde de la pantalla.

En el momento en que se crearon los bordes, se tenía poco dominio de las colisiones, así que para lograr la colisión ocurre atraves de la aproximación. Cada elemento (acorde a su tamaño), tiene definido una distancia la cual al acercarse a un borde y esta es sobrepasada, se entenderá como que ocurrió una colisión y se ejecuta el cambio de posición, manteniendo los mismos valores. Por lo tanto se crea el efecto de salir por un lado y aparecer por el otro.

Hay un pequeño bug en los bordes debido a falta de conocimiento, pues los bordes se ubican a una distancia fija y no son responsivos con la pantalla, cosa que sería lo ideal, por lo que puede que en ciertas pantallas los objetos no han terminado de salir por el borde cuando ya aparecen del otro lado.

4. Efectos

4.1. Fire Explosion

Tanto los efectos de explosiones como el fuego del propulsor fueron códigos extraídos de ejemplos por internet, se dejará referencia de ellos, sin embargo, al igual que con los Asteroides se adaptaron al juego y se extrajo solo lo necesario para que funcionara.

4.2. Sonido - SFX

Se creo una clase SFX, que hace la función de reproductor de sonido. Se cargan los sonidos respectivos al inicio del juego y se tienen funciones básicas para controlar el sonido, como el play, pause, stop, volume, loop.

En cuanto a los sonidos de las explosiones y disparos tienen un tratamiento diferente. Pues al estos sonidos pueden ser reproducidos varios veces en un tiempo relativamente cercano. Para ganar independencia entre ellos y no tener que esperar que un sonido se reproduzca y ejecutar el otro una vez que el primero finalice, se cargan 10 instancias del mismo efecto de sonido, por lo que al ejecutar el sonido, intentará ejecutar el primero, si este se encuentra en reproducción intentará con el siguiente y así sucesivamente y acorde vayan finalizando los sonidos se irán desbloqueando. De esta manera si el jugador dispara muy seguido o explotan muchos elementos se reproducirán los sonidos individualmente.

5. Conclusión

Ha sido un proyecto desafiante y divertido, pues nunca he realizado proyectos en estos lenguajes y tuve que aprender a encontrar soluciones tanto para la parte visual como para la lógica. Considero que aprendí un montón haciendo este proyecto y me hubiera gustado agregarle más cosas, como un registro del puntaje de los jugadores, un mejor menu, agregarle más efectos, añadirle diferentes dificultades e incluso agregarle como objetos booster para mejorar características de la nave, además me hubiera gustado crear un juego con otra perspectiva, pero en vista de la falta de conocimiento y al estar solo, preferí crear una versión básica del juego. A pesar de las dificultades, este proyecto me ha hecho analizar bastante y ha logrado que me interese mucho más por la computación gráfica y me alegra haber disfrutado esta pequeña probada de algo tan maravilloso.

6. Bibliografía

- https://threejsfundamentals.org/threejs/lessons/threejs-fundamentals.html
- https://www.udemy.com/course/introduccion-threejs-3d/
- https://www.youtube.com/watch?v=VxAH-c1ueR4list=PLqhmS_S1oOPHMTxA6Ms67C29XMnz-HbEw
- https://stackoverflow.com/questions/63240082/how-to-get-the-boundingsphere-for-a-whole-scene-in-three-js
- https://github.com/mattatz/THREE.Fire
- https://codepen.io/Divyz/pen/VPrZMy
- https://codepen.io/Xanmia/pen/DoljI
- https://www.youtube.com/watch?v=0Gc_m7Qoxbc