

The Wrath Of The Sea



Integrantes

- Baiges, Matías Sebastián - 59076
- Britos, Nicolás - 59529

Motivación

La idea del juego surgió de hacer algo que sea fácil de jugar, fácil de entender, que sea divertido, y progresivamente más difícil, pero con una dinámica rápida. Es decir, la idea es que te encuentres en un entorno tranquilo en principio, pero con una situación que se avecina que haga que el jugador tenga que enfocarse cada vez más en el juego.

Descripción

El jugador se sitúa en un bote, en un mar infinito, bajo un cielo estrellado en medio de la noche. Si quiere sobrevivir a la furia implacable del mar, deberá esquivar los meteoritos que se le avecinan. Los mismos caen desde todas direcciones del cielo, y una vez impactan

sobre el jugador, el mismo pierde y la escena se reinicia. Existe un contador que contabiliza cuántos meteoritos sobrevivió el jugador, y de esta forma el jugador puede saber si mejoró respecto de la partida anterior.

El juego es compatible con los Oculus Quest 2. El bote puede acelerarse utilizando el trigger del mando derecho, y rotar hacia los lados con el thumbstick del mando izquierdo.

Tareas

El desarrollo del juego se realizó de manera asincrónica, aunque fuimos poniendo algunas ideas en común y mostrándonos el progreso compartiendo pantalla en Discord. La división de tareas se dio de la siguiente manera:

- Matías
 1. Modelado del bote en Blender y skybox
 - a. Para el skybox, se utilizó un generador gratuito [1].
 2. Mar y oleaje
 - a. Para el mar, se creó un script que crea chunks de agua alrededor del player.
 - b. Para el oleaje, se modificaron los vértices de la malla del plano de agua, utilizando un controlador para las olas, que se encuentran generadas por la suma de 3 funciones $y(x,z)$ (las mismas se determinaron empíricamente, es decir, viendo los resultados que se iban formando):

```
private float Wave1(float _x, float _z) {  
    float offset = Time.time * speed1;  
    return amplitud1 * (Mathf.Sin((_x + Time.time) / length1 +  
offset) + Mathf.Cos(_z / length1 + offset));  
}  
  
private float Wave2(float _x, float _z) {  
    float offset = Time.time * speed2;  
    return amplitud2 * (Mathf.Sin((_x+_z) / length2 + offset));  
}  
  
private float Wave3(float _x, float _z) {  
    float offset = Time.time * speed3;  
    return amplitud3 * (Mathf.Sin(_z / length3 + offset));  
}
```

3. Buoyancy (Flotabilidad)

- a. Para que el bote tuviera un efecto realista, basándonos en un video explicativo [2], agregamos flotabilidad al player. Dadas las características de nuestras olas, tuvimos que modificar hasta ajustar a gusto el drag (arrastre) del Rigidbody del player, para que quedara lo más realista posible.

4. Meteoritos

- a. Lo primero fue instanciar los meteoritos en un entorno (dirección aleatoria) del jugador, a una distancia especificada del mismo, y a una altura aleatoria.
- b. Los mismos se instancian con rotación y escalado aleatorios, lo cual diversifica los meteoritos presentes en la escena.
- c. Luego, se calcula la velocidad a la que deben ser enviados para conseguir una trayectoria que acierte en una zona próxima al jugador (un poco más adelante generalmente pues se espera se esté moviendo).

● Nicolás

5. Sonidos

- a. Sonidos inmersivos de ambiente, como ruido de viento/océano en loop o sonidos cuando pasan los meteoritos al lado del jugador.
- b. Sonidos interactivos, como el ruido del motor del bote, que se va ajustando en base a la aceleración del mismo, sonidos que se emiten cuando se generan nuevos meteoritos y cuando caen al agua. Para esto, se fueron mezclando qué tan “2D” o “3D” es el sonido, así como la distancia a la cual el mismo puede ser escuchado. De esta manera, el jugador puede sentirse más inmerso en la realidad del juego.

6. Dificultad progresiva


- a. Se implementó que con el paso del tiempo (mientras más meteoritos se sobreviven), la velocidad a la que se dirigen los meteoritos es cada vez mayor (llegando en menor tiempo al jugador), lo cual obliga al jugador a estar más atento y haciendo el juego más divertido.

7. Efecto de partículas en meteoritos

- i. Se implementó un sistema de efecto de partículas sobre el meteorito para brindar luminosidad y provocar una mayor sensación de inmersión.

Recursos utilizados

[1] - <http://www.tyro.github.io/planet-3d/>

[2] -  How to Set Up Dynamic Water Physics and Boat Movement in Unity | Ship Buoy...