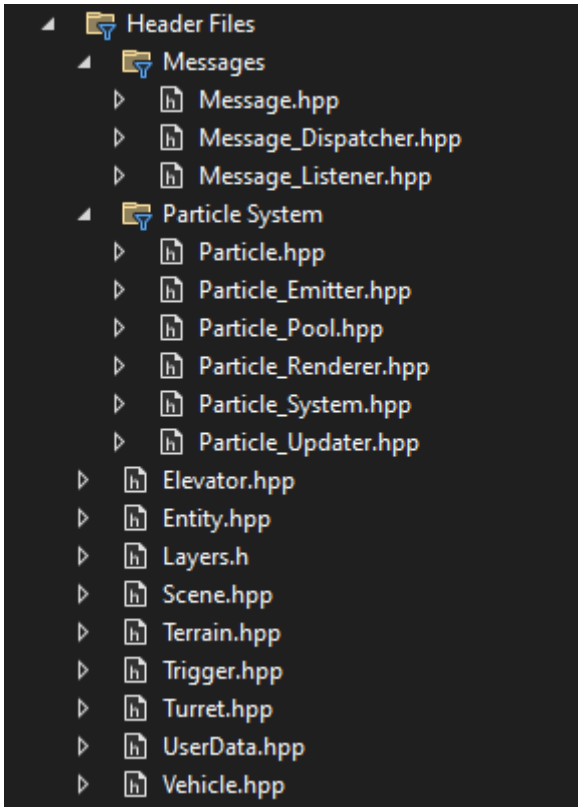
Documentación Práctica 1 Animación 3D



Para esta práctica se ha optado por una composición de escenas y entidades.

La escena se encarga de renderizar entidades, contener y simular un mundo de físicas, escuchar por contactos entre entidades, recoger input y mandar mensajes entre entidades. Cada entidad se guarda en la escena con identificador único, y para que pueda ser renderizados necesitan añadir una serie de cuerpos a su lista de b2Bodies.

Las entidades son objetos que pertenecen a una escena, contienen una serie de parámetros que les permiten simular físicas, renderizarse en caso de que fuera necesario y actualizar su lógica interna en tiempo de ejecución

Si una entidad requiere de una configuración previa de sus parámetros ya sea para visuales o lógica se deja a su disposición un método **initialize**, el cual es llamado por la escena cuando esta se crea y comienza su ejecución.

Para sus comportamientos se deja también un método **update** en caso de que dicha entidad requiriera de una actualización o comprobación constante de alguna lógica. Estos métodos update también se llaman desde la escena en su bucle principal de run.

De la clase de entidad heredan las clases de:

- **Terrain**: Esta clase contiene todos los elementos **estáticos** que componen el decorado de la escena, no contiene lógica por lo que no necesita de un método update.

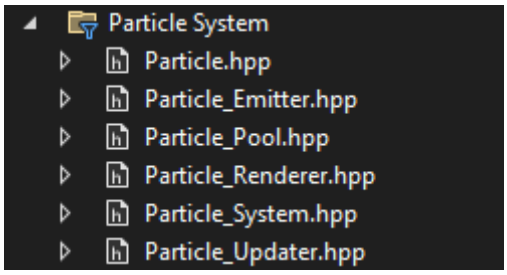
- **Vehicle**: El vehículo es el objeto que el jugador controla usando las teclas propuestas de A y D para movimiento, espacio para frenar y V para abrir o cerrar el maletero. Se desplaza por el escenario usando los motores de Box2D y recogiendo el input en el método update que deriva en funciones privadas que manejan dicho input.
- **Turret**: La torreta es la entidad encargada de guardar las bolas que tendrá que usar el jugador para completar el nivel. Esta esta compuesta por un par de bodies unidos por una joint. Es a su vez también un Message_Listener, lo que significa que puede recibir mensajes que evoquen respuestas concretas en base al contenido del mensaje. Esto es usado para que cuando el jugador haya llegado a la derecha de la pantalla la torreta gire, soltando las bolas de su parte superior.
- **Elevator**: El ascensor es un componente que no se encuentra instanciado desde el comienzo, solo en el momento en el que el jugador consigue llegar al final de la escena. Es un componente que contiene un trigger interno ya que la lógica es que siempre que haya un ascensor es necesario que haya algo que le diga cuando subir o bajar por lo que, si no habría que instanciar un trigger externo siempre que hubiera un ascensor, no se movería con él y no se podría manejar su comportamiento tan exactamente como si estuviera dentro ya de la propia clase. En el momento en el que el jugador se sube en él comienza su ascenso hasta un punto concreto.
- **Trigger**: El trigger es una clase destinada a ser repetida a lo largo de la escena, admiten un parámetro de id que será el mensaje que envían una vez un objeto interacciona con ellos por medio del Message_Dispatcher. Los trigger no añaden sus bodies a su lista, por lo que no se renderizan a menos que se indique lo contrario.
- **Particle System**: El particle system se hablará en más profundidad algo más adelante, pero se considera entidad para poder actualizar sus métodos desde la escena con el resto de elementos y actualizar posición de las partículas.

Para los mensajes se utiliza como se ha comentado un sistema de mensajería clásica junto con el sistema de detección de colisiones de Box2D.

El sistema de colisiones y **ConctactListener** se gestionan desde dentro de la escena, ya que no habría mucho sentido en tener esa clase fuera de la escena al solo existir realmente dentro de la misma. Esta clase escucha posibles contactos entre dos cuerpos, si las colisiones contienen sensores (triggers) se envía un mensaje en base al user data guardado dentro de dichos bodies si es que tuvieran algo. Cuando se detectan colisiones útiles se envían mensajes a las correspondientes entidades que estuvieran interesados y ya ellas manejaran la colisión como sea necesario.

Para el **Particle System** se sigue el esquema descrito en clase de un sistema de partículas que contenga un emisor, un updater, renderer y particle pool. Se crean en la escena usando unos parámetros pasados por constructor para configurar su posición y cantidad de partículas y reciben mensajes para poder comenzar a emitir

partículas cuando una de las bolas entra dentro del trigger final en la plataforma flotante.



Se renderizan usando SFML aunque lo ideal sería tratar de usar la GPU.