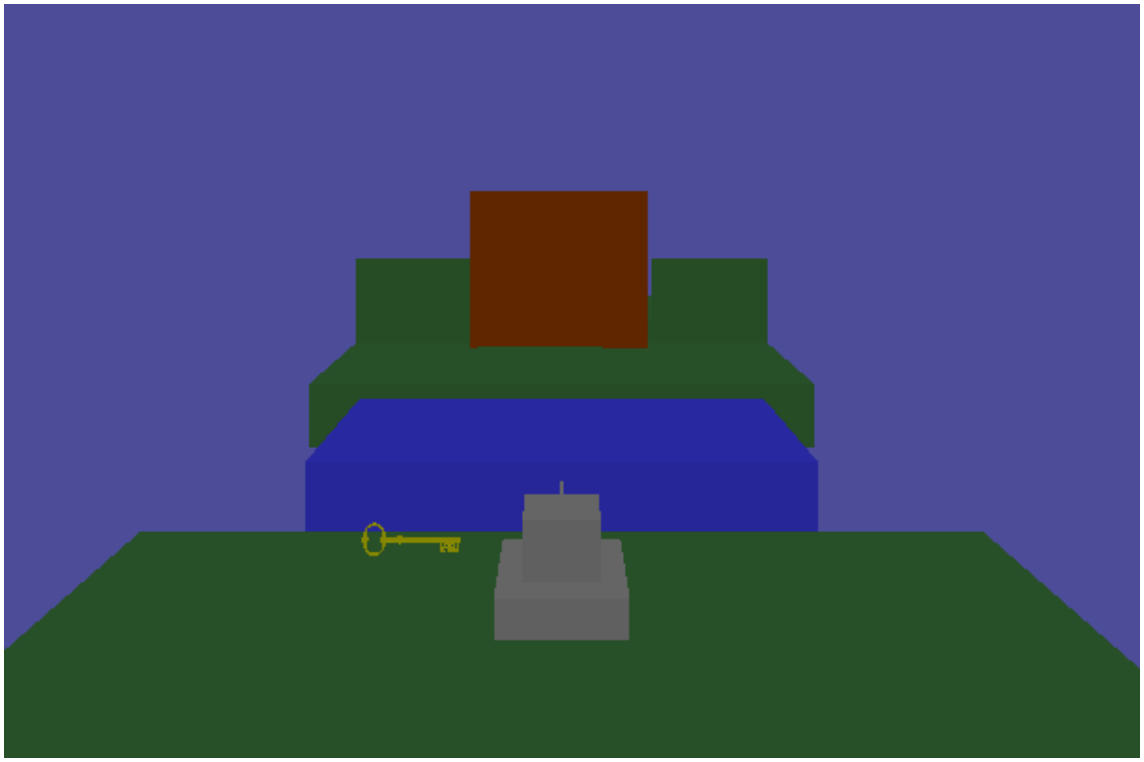
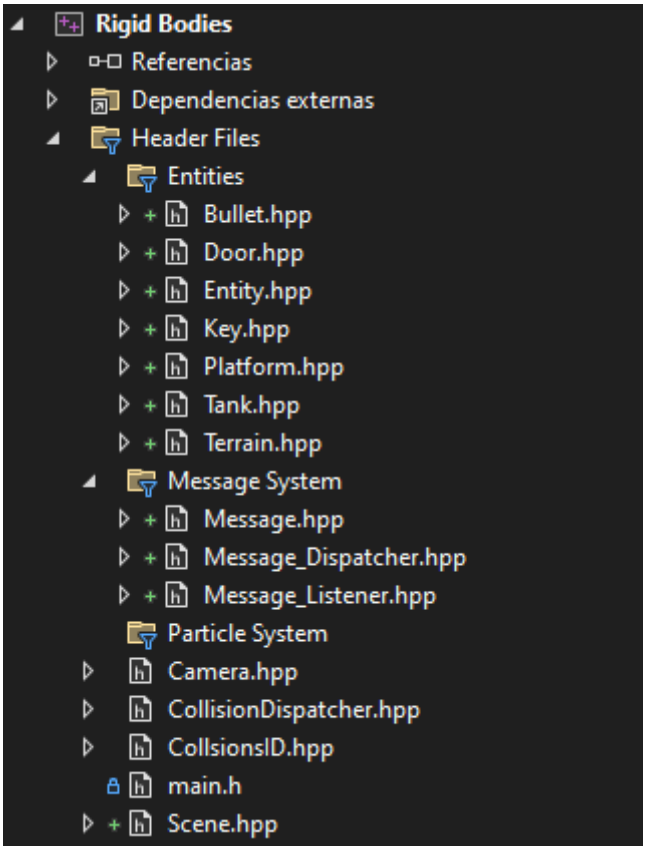
Documentación Práctica 2 Animación 3D





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 **btRigidbodies**.

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. Se guardan una serie de punteros a los componentes importantes como los rigidbodies, motionStates y collisionShapes para todas las entidades y por ejemplo, para el tanque, se guardan también referencias a las constraints.

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. A su vez también contiene la pila de cubos dinámicos del fondo que el jugador puede derribar con la torreta.
- **Bullet**: La clase bullet crea un objeto dinámico que se instancia con un impulso frontal en la dirección en la que está apuntando el cañón. Al cabo de un rato las balas despawnean de manera forzada, sería mucho mejor usar una pool de balas para su manejo.
- **Key**: La llave contiene un objeto estático con una ID de colisión dentro del UserIndex. Escucha mensajes y cuando el Conctact Listener detecta que han colisionado tanque y llave manda un mensaje customizado y la llave desactiva sus colisiones y sus visuales
- **Platform**: La plataforma es un objeto que viaja constantemente entre dos posiciones de valores de X distintas, moviéndose de manera lineal. No fui capaz de hacerlo funcionar con velocidades lineales.
- **Tank**: El tanque es el componente principal de la escena. Contiene una serie de rigidbodies, chasis, torreta y el cañón. El chasis al no tener movimiento el tanque se ha quedado estático. La torreta está pegada al chasis con una bisagra y el cañón a la torreta con otra. Rotan usando dos motores para girar sobre sus ejes. Para el disparo cuando el jugador pulsa la Q se genera una bala con una fuerza frontal en dirección de donde apunta el cañón. El movimiento del tanque por algún motivo da algún problema de inputs, de vez en cuando es como que se queda pillado en una posición dada.

Para los mensajes se utiliza como se ha comentado un sistema de mensajería clásica junto con el sistema de detección de colisiones de Bullet. Se guarda un message dispatcher y se suscriben entidades que quieran escuchar mensajes, desde el ContactListener se lanzan mensajes y los que tengan cierta ID los reciben.

El sistema de colisiones y **ConctactListener** se gestionan desde un sistema aparte que hace referencia al contact dispatcher que contiene el mundo físico de Bullet creado al comienzo de la escena. Esta clase escucha posibles contactos entre dos cuerpos, si las colisiones contienen user index guardado dentro de dichos bodies se comprueba que tipo de User Index. 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 manejo de modelos 3D, al no poder manejar las escalas directamente desde Bullet, y los métodos presentados para tratar de manejar sus dimensiones de manera manual me parecían demasiado complicados para lo simple que debería de ser algo como importar modelos a un programa de físicas, opté por añadir los modelos a Blender, pasarlos a escala 1,1,1 y luego reimportarlos a la aplicación. Entonces, desde ahí, le aplicaba la escala directamente al rigidbody que era la que usaba el modelo para renderizarse, entonces había una concordancia entre visuales y físicas sin muchos problemas o errores.

También, hay veces que me salta una especie de punto de parada al cerrar la aplicación, el cual no es como tal un error ni un warning, pero no entiendo por qué sale, no conseguí solucionarlo, entonces a veces cuando se cierra la ventana desde la cruceta ocurre.

Para los archivos 3D que se muestran en el programa, llave y puerta, se han usado estos dos modelos gratuitos de sketchfab:

- Llave (Autor Yomans):
    - https://sketchfab.com/3d-models/key-7a0f6aaffe604d65bb560955990ce68b.

- Puerta (Autor Mehdi Shahsavan), No se ve mucho debido a la luz, pero el modelo está añadido:
    - https://sketchfab.com/3d-models/door-metal-door-window-10c8a0789ed3474baf53b8a185809cb7

Si se me permite. Como conclusión, he de decir que al igual que trabajar con Box2D fue bastante cómodo y agradable, supongo que será por mi falta de experiencia, pero trabajar con Bullet ha sido extremadamente frustrante y desagradable. No hay documentación, los foros son respuestas de gente sobre complicando las cosas, las acciones más simples están expuestas de manera compleja, no hay acceso a funciones o parámetros clásicos disponibles para trabajar… No es una librería con la que tenga ganas de volver a trabajar en ningún momento.