TANQUETA. PROGRAMACIÓN VISUAL PARA VIDEOJUEGOS.

Memoria. Víctor García Cortés.

1.- Introducción.

"Tanqueta", práctica final de la asignatura Programación visual para videojuegos (PVV), consiste en el diseño de un vehículo de 6 ruedas en el cual estará implementado un cañón, todo ello controlable por el usuario, mediante el cual se podrá disparar y destruir objetos.

El vehículo está basado en la utilización del WheelCollider. Wheelcollider es una técnica de unity que permite la simulación de ruedas de vehículos, las cuales actuarán como tal, haciendo mover el vehículo. En este caso utilizaremos un mínimo de 6 ruedas, un wheelCollider por cada una, como si de una tanqueta se tratara. A través del script, podremos accionar cada wheelcollider con un comportamiento diferente, podremos dar a algunas ruedas la acción de ruedas motrices y a otras ruedas directrices, de esta manera unas ruedas se encargarán de la dirección y otras ruedas se encargarán de la velocidad, o de ambas a la vez. Pudiendo regular estos parámetros en todo momento.

El jugador podrá manejar a través de las teclas de dirección o a, w, s, d, tanto la velocidad como el movimiento de giro. A su vez, con el ratón podrá controlar la dirección del cañón, de esta manera apuntará con el ratón donde queramos disparar.

La acción de disparar se hará con el botón izquierdo del ratón. Esta es la manera más sencilla de controlar el vehículo y disparar a la vez.

Al disparar, se instanciará un gameObject, el cual saldrá disparado, simulando la acción de disparo. Este gameObject podrá interactuar con los elementos de la escena, pudiendo por ejemplo romperlos, para hacerse paso.

2.- Descripción Técnica.

Análisis funcional:

1. Primer paso: creamos los scripts que nos permiten montar un vehículo personalizable "VehicleController02". Este script será añadido al vehículo creado en la jerarquía, en el inspector indicaremos el número de ruedas que queremos que tenga el vehículo y cuáles van a ser directrices y/o motrices. El vehículo estará formado por la cabina, donde se encontrará la torreta, el motor, donde incluiremos tanto las ruedas físicas como las visuales, estas últimas solo sirven para simular que las ruedas se mueven, pero las que hacen todo el trabajo son las físicas y por último el centro de gravedad (COG), esté es muy importante ya que dependiendo de la posición en la que se encuentre, repercutirá en el manejo del vehículo.

2. Segundo paso: montamos la tanqueta, esta tendrá un total de 6 ruedas, en mi caso las dos primeras serán directrices y el resto motrices.

3. Tercer paso: una vez que tenemos el vehículo, con el peso y velocidad ajustados. Pasamos a hacer la torreta que irá en el centro del vehículo.

4. Cuarto paso: La torreta está compuesta de dos cilindros, un total de 3 partes. Cada una de ellas hija de la anterior. Torreta > Cannon Pivot > Cannon. Torreta tendrá el script del movimiento donde controlaremos la dirección tanto en el eje “x” como en el “y” del cañón, todo ello a través del ratón.

5. Quinto paso: Incluir la mecánica de disparo del cañón. Esto se hace a través del script “ShotController”. Por medio de este script, simularemos la mecánica de disparo del cañón, esto se hace instanciando un gameobject cada vez que pulsamos el botón izquierdo del ratón, en la posición del tubo del cañón. Una vez que se ha instanciado el objeto se le añade una fuerza que simula el efecto de disparo. Por medio de este script para que la simulación sea un poco más realista, hacemos uso de una corrutina en la cual accederá a un animation curve con el cual le daremos un efecto de retroceso a la cámara principal, cada vez que se accione la mecánica de disparo.

6. Por último a través de los script “Health” y “BulletController”, añadiremos vida a objetos de la escena, y daño a las balas. El script “Health” añade vida a los objetos que contengan el script, y a través del script “BulletController” el cual se ha añadido al prefab del gameObject instanciado al disparar, añade un daño a la bala, si esta impacta con otro objeto, le restará vida, si esta es igual o menor que cero, se destruirá el objeto. Tanto la vida de los objetos como el daño son totalmente editables, pudiéndose cambiar en cualquier momento.

Diagrama de clases: Casos de uso. Nombre de los scripts: **VehicleController02.cs, TurretController.cs, ShotController.cs, Health.cs, BulletController.cs.VehicleController02.cs, TurretController.cs, ShotController.cs, Health.cs, BulletController.cs.**

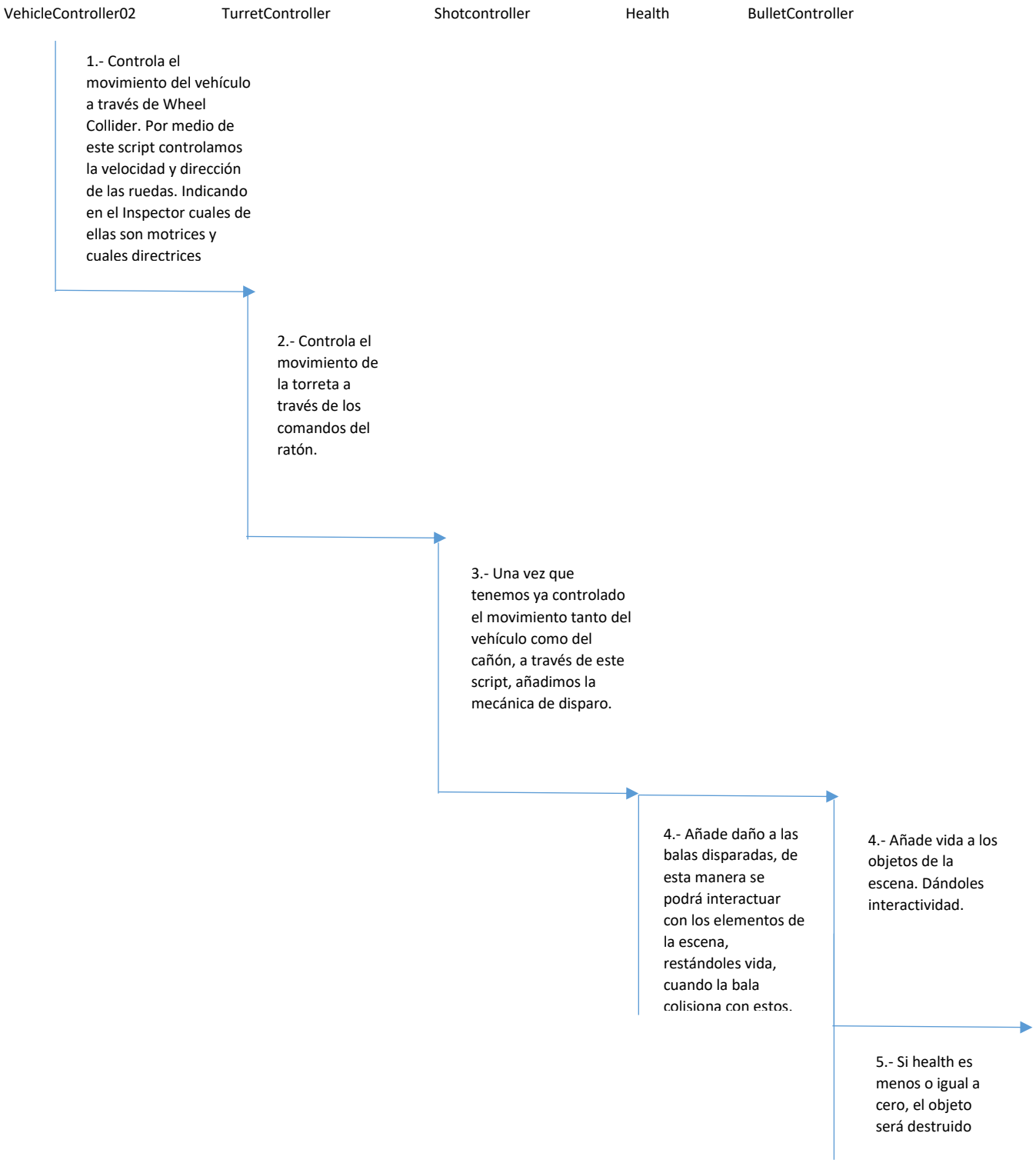
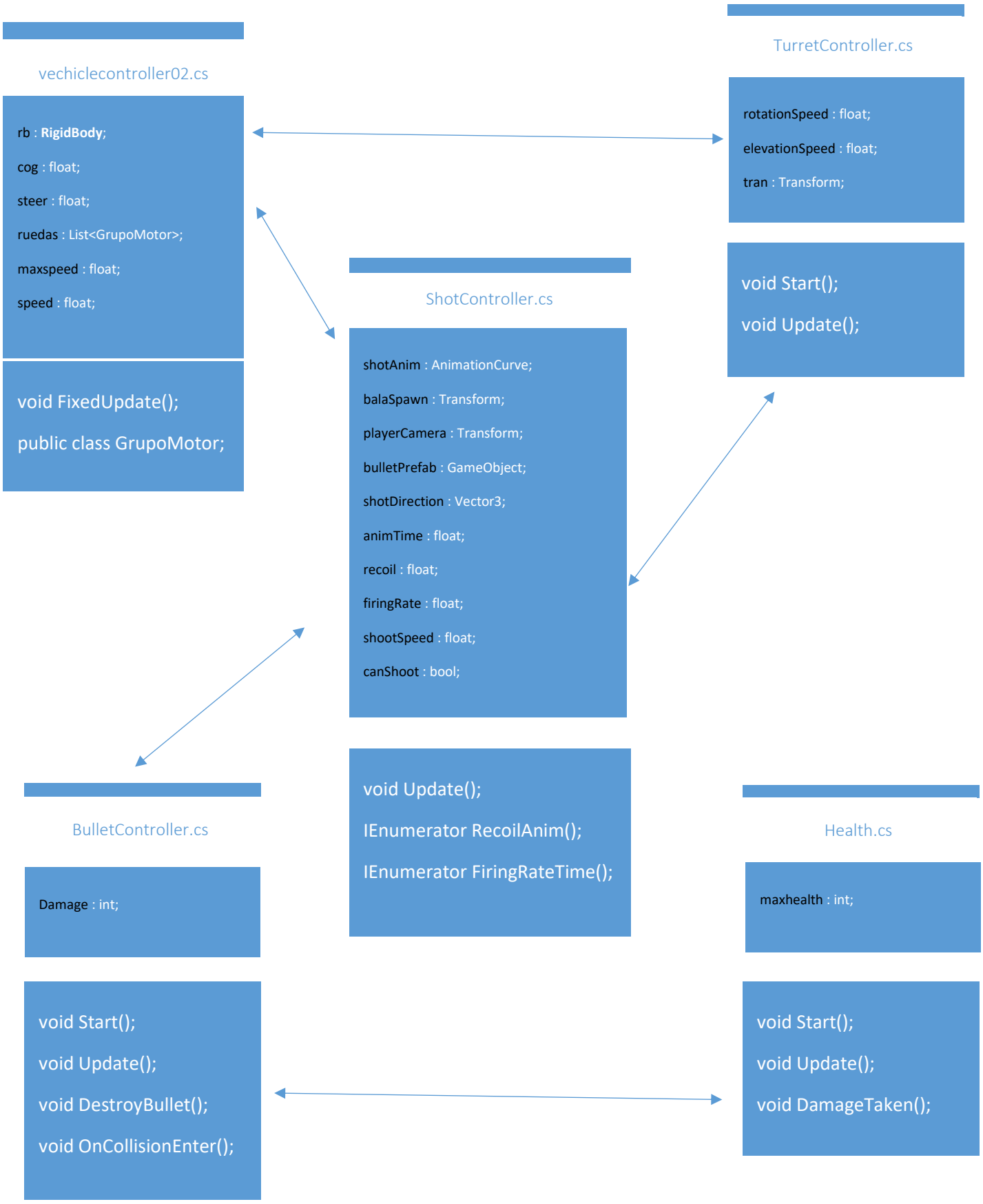


Diagrama de clases:



3.- Diario de desarrollo.

A lo largo del proceso he encontrado varias dificultades, una de ellas fue, añadir el movimiento a la torreta a través del ratón, el problema no fue en el script, si no que fue en la jerarquía, a la hora de posicionar jerárquicamente los objetos y saber si es el padre el que debe contener la malla, quién debe llevar el collider, etc…, esto puede suponer un gran problema porque normalmente siempre pensamos que el fallo será de código y te obcecas en él poniéndolo de diferentes maneras, todas con el mismo resultado. El fallo que tuve es que en la torreta, en la parte llamada cannon pivot, yo tenía ahí el mesh, el cual debe estar en el padre (Torreta), fallo que tras enseñárselo a un compañero de clase (Jesús Villar) me consiguió solucionar. Este fue un problema no relacionado con el código, cosa que no le quita importancia a la hora de que el proyecto funcione correctamente.

Otro problema que tuve fue con la restricción del movimiento que tiene que hacer el cañón. No conseguía restringir los valores máximos que debe tener este. Lo cual repercutía negativamente en el disparo.

Otro apartado con lo que tuve dificultades fue al añadir la curva de animación a la cámara, intenté hacerlo por mi cuenta, pero no me queda muy claro el funcionamiento y la estructura de su programación, al no conseguir hacer compilar el código, opté por no incluirlo en el proyecto. Como resultado tuve que utilizar el ejemplo que nos enseñaste en clase. Por tanto en este proyecto la cantidad de dificultades se han visto reducidas pero en contraposición la cantidad de aprendizaje ha sido mucho mayor.

Por tanto el balance total en este proyecto ha sido mucho mayor el aprendizaje que las dificultades encontradas lo cual, es muy gratificante.

Por otra parte he aprendido muchas cosas, es un proyecto bastante completo, en el cual se aplican muchos conocimientos vistos este curso, lo cual es muy beneficioso ya que te ayuda a afianzar lo aprendido y gracias a ello también te das cuenta de todo aquello que no llevas tan bien y conviene repasar. En mi caso destaco las curvas de animación.

Todo lo relacionado con los WheelCollider a la hora de hacer vehículos me pareció muy interesante y lo que conseguimos hacer es un gran avance, solo pensar que hemos visto una mínima parte de todo lo que abarca, me parece algo muy positivo ya que nos has dado las bases para seguir con ello y aprender muchísimo acerca de vehículos.

Como conclusión me gustaría decir que ha sido una asignatura que está muy bien planteada, nos has enseñado la base de muchas técnicas de programación, de estructuración de código y una gran cantidad de conocimientos de unity. Gracias a todos los proyectos que hemos hecho, los cuales están muy bien enfocados, nos has dado un gran base, la cual nos va a servir para ampliar los conocimientos adquiridos y continuar con ellos para hacer nuestros propios proyectos, o seguir mejorando aquellos que hemos realizado durante el curso. Muchas gracias Luis.

Víctor García Cortés.