**Ejercicio Redes – Andrés Pérez**

La implementación del disparo en red se ha realizado de la siguiente manera.

1º Creación de un input action que permite a los clientes mandar un mensaje que notifica al servidor de que ha de crear una bala. Este mensaje es del tipo nuevo BULLET\_SPAWN y manda como dato extra en el paquete el ID del dato al que va a pertenecer la bala.

2º Creación de ABullet: Este actor es la bala, contiene un collider, una static mesh y el component de net de la bala. Se encarga de actualizar la posición cada frame de la bala tanto en cliente como en servidor, y en el caso del servidor también se encarga de chequear colisiones con los coches. Se decidió que no se mandarían paquetes de la posición de la bala y en su lugar los clientes actualizan por su cuenta dicha posición. Esto se hizo porque al ser un movimiento que no cambia de condiciones (siempre misma velocidad y dirección) la posición en Cliente y Servidor casi nunca va a ser distinta y por lo tanto para reducir el volumen de paquetes que se mandan se decidió que esta sería una opción de optimizar que no conlleva un riesgo alto.

3º Creación de UBulletNetComponent: Este componente se encarga de Destruir la bala en el servidor (pasado un tiempo o al colisionar) y mandar el mensaje a los clientes para que estos también destruyan sus representaciones de la bala. Además, se encarga de lanzar los mensajes de colisión cuando la bala le notifica de que ha entrado en contacto con otro actor.

4º Cambios en el GameNetMgr: Se crearon 2 tipos nuevos de mensajes (BULLET\_SPAWN, BULLET\_DESTROY):

* BULLET\_SPAWN: Este mensaje se encarga de crear una bala y en el caso del servidor mandar un mensaje a los clientes para que creen una bala que represente la del servidor. Recibe un paquete cuyo único dato extra es el cliente encargado de la bala.
* BULLET\_DESTROY: Este mensaje solo lo reciben los clientes y sirve para notificarles de que tienen que destruir una bala específica. Recibe un paquete cuyo único dato extra es el ID encargado de la bala.

Por lo tanto, el flujo de mensajes queda de la siguiente forma:

**Creación de la bala**

CLIENTE: Pulsa Espacio -> Manda Paquete (BULLET\_SPAWN, ID).

SERVIDOR: Recibe Paquete (BULLET\_SPAWN, ID)-> Crea Bala -> Manda paquete (BULLET\_SPAWN, ID)

CLIENTE: Recibe Paquete (BULLET\_SPAWN, ID)-> Crea Bala

A partir de este momento la bala actualizara su posición en servidor y cliente sin necesidad de mandar mensajes.

**En caso de colisión (Lógica en servidor)**

SERVIDOR: Manda Paquete (BULLET\_DESTROY, ID) -> Destruye Bala y para al Coche impactado

CLIENTE: Recibe Paquete (BULLET\_DESTROY, ID) -> Destruye Bala

**En caso de que no haya colisión (Lógica en servidor)**

SERVIDOR: Pasados X segundos manda Paquete (BULLET\_DESTROY, ID) -> Destruye Bala

CLIENTE: Recibe Paquete (BULLET\_DESTROY, ID) -> Destruye Bala