**Actor**: tiene lógica pero no se mueve (no tiene control, se puede programar para que se mueva pero no recibe input)

**Pawn**: puede recibir input de un controlador (no tiene por que ser personaje)

**Character**: malla esqueletal, es lo mismo pero le metes un motor de animaciones (ya hay un señor que camina) 🡪 aquí están las **IAS**.

**Player** **controler**: es un player específicamente 🡪 aquí está el **jugador**.

De normal para nuestro player vamos a coger uno que esté hecho y a partir de ahí lo modificamos.

Si fuera en C++ y le hacemos doble click abriría el visual studio para modificarlo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Con las blueprints podemos cambiar:

Interfaz de usuario gráfica

Descripción generada automáticamente

* **MaxStep Hight**: máximo de altura que podemos subir sin saltar
* **Gravity scale**: afecta a cuanto salta el player
* Tamb está si puede nadar o no, esto lo podemos cambiar.

También podemos cambiar la cámara para que se parezca mas a esto

Imagen que contiene interior, hombre, cuarto, tabla

Descripción generada automáticamente

Nos vamos al Viewport y la movemos.

Imagen que contiene Diagrama

Descripción generada automáticamente

## Action Mappings

Vamos a detectar pulsaciones de teclas. Y será un mapeo

Por ejemplo que la Z dispare.

Edit > Project Settings (para cuando salga en el Unreal Store)

* Descripcion del juego
* Nombre del juego ()
* Thumbnail: icono
* Nombre de la compañía

En el Engine

Aquí es donde decimos si queremos sombras o no, con mas o menos calida, etc.

Si buscamos AcTION en el buscador y vemos que podemos ir al

Engine > Input

Y vemos que hay un Action Mappings en Jump

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Esta creada una acción que es Salto y está mapeada al SPACE en PC o al Gamepad

Añadimos una Shoot (action) y le ponemos la M. Ahora NUNCA hablamos de la M, decimos del OnShoot o la acción Shoot

Si ahora vamos al ThirdPersonManager, a la Blueprint y hacemos click derecho podemos verlo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Objeto Mina, que sea cilindro como un disco y que pueda instanciarlas.

Ojo no puedo enchufar uno al otro porq lo de la izq me devuelve un actor y a la derecha necesita un componente. Es decir el personaje lo fabrica y ella sola se tendrá que mover

Interfaz de usuario gráfica

Descripción generada automáticamente

Para quitar las colisiones con la granada nos camos a los detalles de la static mesh para que no colisiones con el player

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En vez de moverlo con moveComponent creamos un componente que se llama **ProjectileMovement**. Y aquí hay por defecto un montón de parámetros

Arrow: son ejes (o ubicación mejor dicho), Le vamos a meter una arrow (direccion) al personaje, para que cuando se fabrique la granada en esa dirección.

En juegos hay 3 tipos de proyectiles:

* **HitsCan**: Un rifle dispara muy rápido. La bala va muy rápido y puede ser que la bala no colisione y atraviese. Por eso se usa Hitscan. Es una línea. (tipo Call Of Duty). Ni si quiera usamos la clase Projectile. Lo que hacemos es una línea y si estas en la línea te he dado (coste 0)
* **No** **Balísticos**: cuando mueves la bala podemos mover por
  + **Cinemática**: V = v \* t BLA BLA BLA, (esto es mediano de caro). Va moviéndose poco a poco. (Coste 2)
  + **Dinámica**: hacemos una integral (muy caro)
* Balísticos:

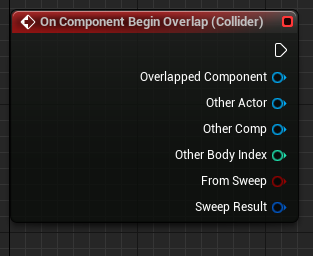
Shotgun: cada disparo son 40 proyectiles.

Projectile gravity scale: 0 (no cae con la gravedad).

Actor que no haga nada, que tenga variable vida y si llega a 0, se muere

Tipos de daño:

* Por contacto: si la bala me toca me hace daño
* Por explosión: si le doy con un lanzacohetes no hace daño como tal si no que explota y me da
* Por barril: yo le doy al barril y el barril provoca explosión que me da a mi.



Cuando hago esto el other actor es lo que choca contra mi

* Modificar la mina para que se convierta en barril explosivo:
  + hacemos que sea mas alto
  + Le ponemos Life (cuando life = 0, lanza una explosion)

Como hacer qie algo que explote haga daño: lo fácil seria meterle un collider de colisión y otro de explosión, pero si algo entra en contacto con el grande te haría daño antes de que explote, no me sirve. Tenemos 2 blueprints: una es el barril y la otra es el área de daño. El área de daño en su begin play va a generar el efecto de explosión y activara el collider.

Podemos hacer el portal, el Mario. Para que el vea que hemos hecho un juego más allá de lo que hemos hecho en clase. En una semana de curro podemos hacer el portal. Incluido el teletransportarse. Youtube está petao de tutoriales. Hay que verlos para

Se entrega después de navidad, justo después de navidad.