Introduction

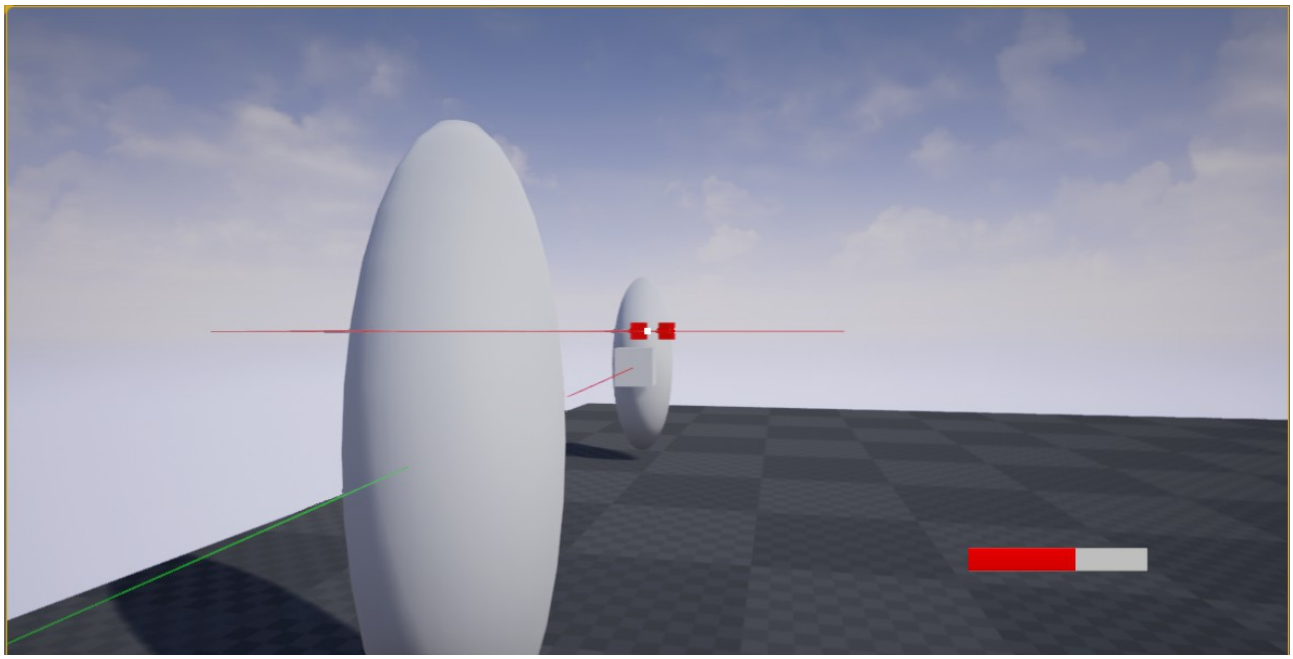
(NB : J'espère que ce tuto vous sera utile et qu'il sera compréhensible par tous. Si vous avez des questions n'hésitez pas à m'envoyer un message sur Facebook « Antoine Gargasson » ou par mail antoine.gargasson@gmail.com)

(NB2 : Ce tuto est réalisé sur la version 4.8.3 de UE4)

Dans le tuto d'aujourd'hui nous allons nous appliquer à créer un système de tir, de dégâts, avec une interface très simple et une IA tourelle qui nous suivra et nous fera des dommages régulièrement.

L'objectif est vraiment de bien comprendre comment fonctionne l'envoi de dommages et la réception de ces derniers en blueprint.

Voici un aperçu du résultat :



Debut

Comme d'habitude, commencez par lancer UE4 et créez un nouveau **projet vide**. Nommez le à votre convenance.

Vous arrivez dans la scène vide (jusqu'à là rien de nouveau). Créez **deux dossiers** dans votre **Content Browser**, le premier se nommera **Blueprint** et le second **UMG**.

Pour une fois nous allons commencer par le **visuel**, vous allez vite comprendre pourquoi.

En **UMG** nous allons donc créer un **viseur** et la **barre de vie** du joueur.

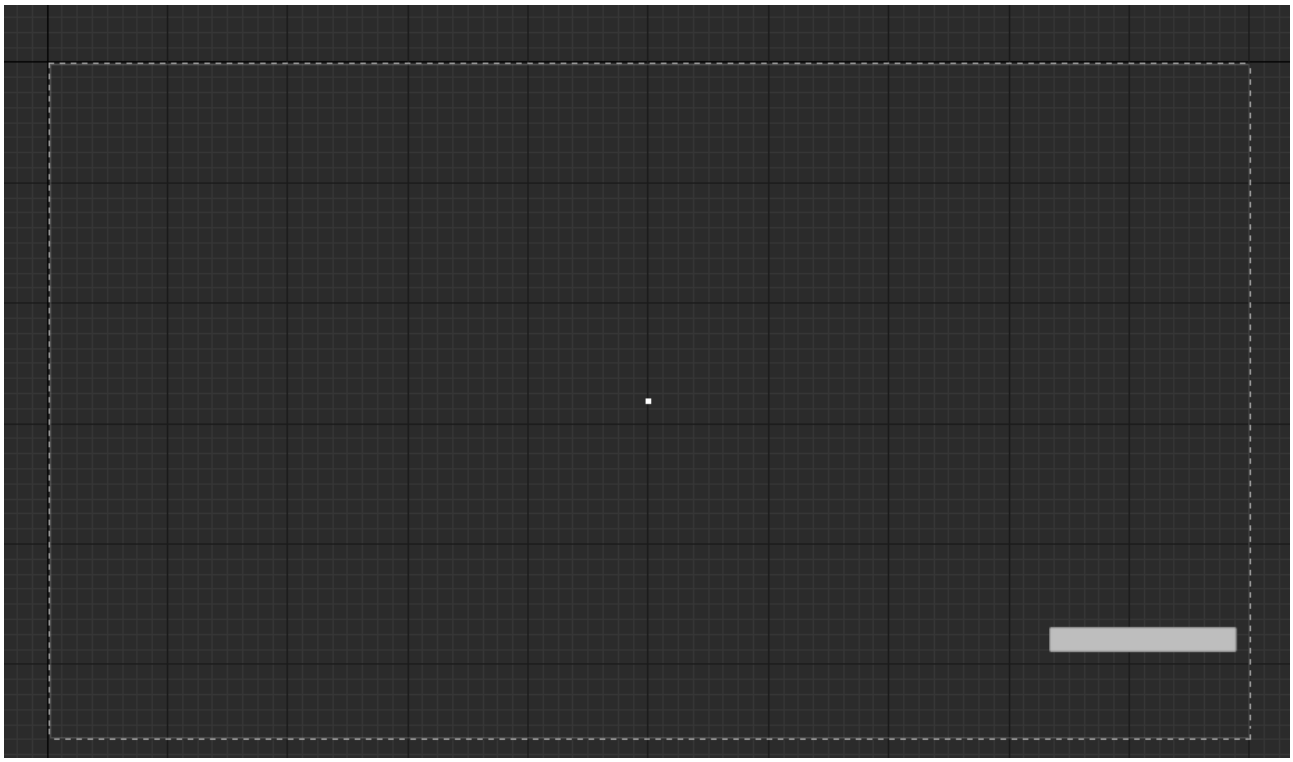
Commencez par créer un **Widget blueprint** (User Interface → Widget Blueprint) et nommez le **UMG_HUD**.

Gardez le **Canvas Panel** et mettez une image. **Redimensionnez**-la en 10 par 10 pixel. **Placez l'ancre** au **milieu** de l'écran (**Anchors**) et **placez l'image** en -5, -5.

Ceci est votre **viseur**.

Ajoutez une **progress bar**, placez la en **1600**, **900**. **Ajustez** la de **taille 300** par **40**. Vous pouvez modifier le **Fill Color and Opacity**, je l'ai mis en rouge ;)

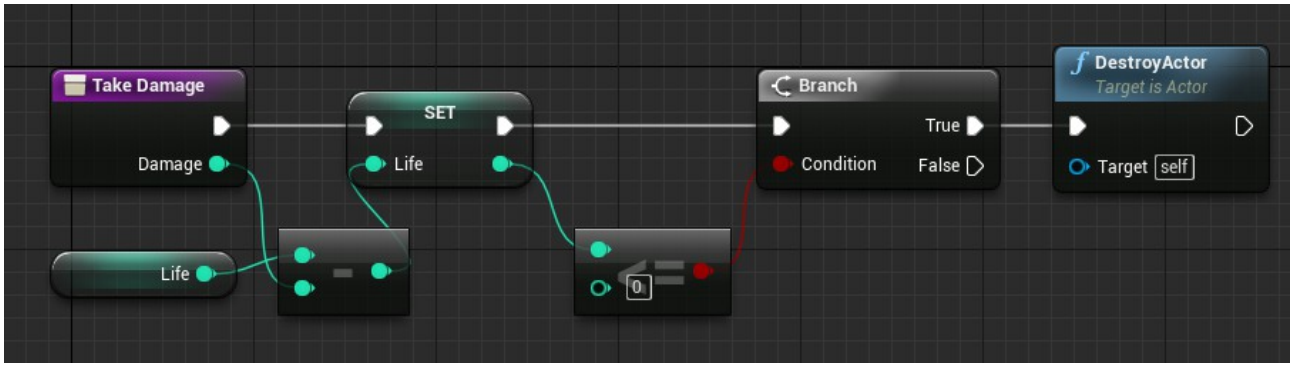
Votre **vie** est **prête**, il ne restera plus qu'à la **binder**.



Fonctionnel

Créez maintenant un **blueprint** de type **character**. Nommez le **BP_Human**. Ce **blueprint** sera la **classe mère** de notre **joueur** et de notre **ennemi**. Ajoutez lui **3 variables** de type **integer**. Nommez les **Life**, **LifeMax**, **Damage**.

Créez maintenant une nouvelle **fonction TakeDamage**. Elle prend en **entrée** un **integer**. Dans cette fonction on va retirer les dommages que l'on reçoit en entrée et vérifier si le personnage est mort.



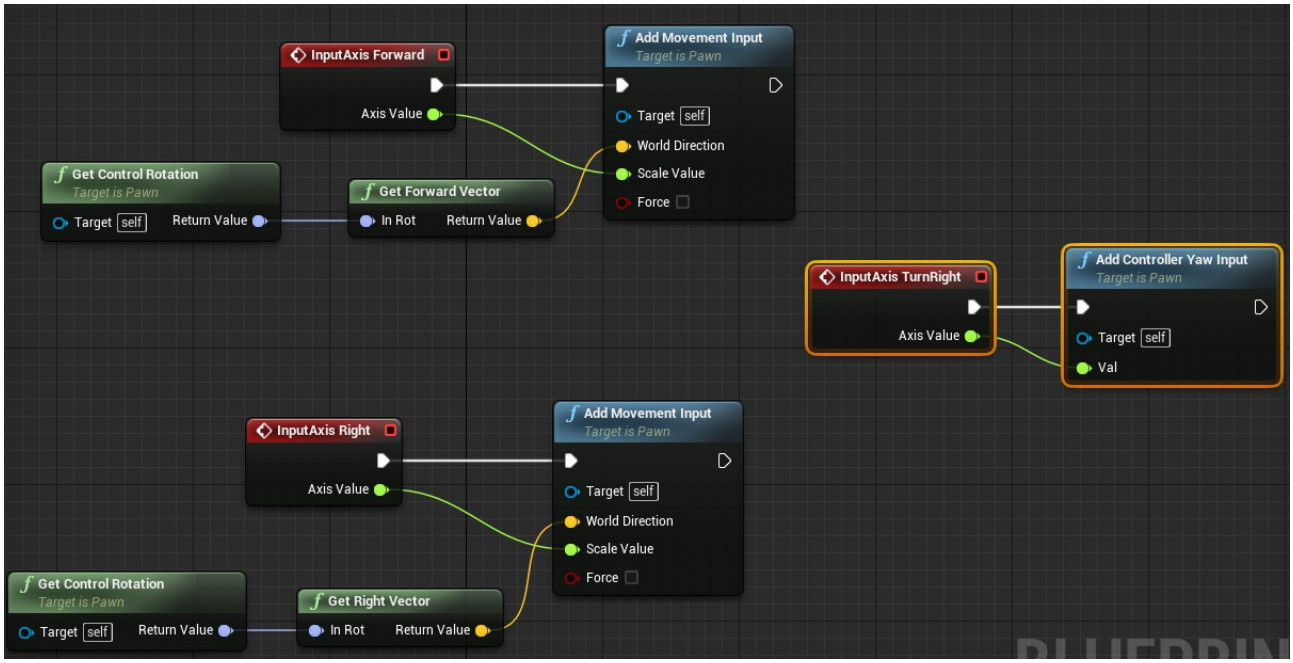
Faites un **clique droit** sur votre **BP_Human** et **create child blueprint class**. Vous allez créer un enfant de cette classe. Nommez le **BP_Player**.

Comme pour le joueur, faites un **create child blueprint class**. Nommez le BP_Enemy.

<u>Passons maintenant au **joueur** :</u>

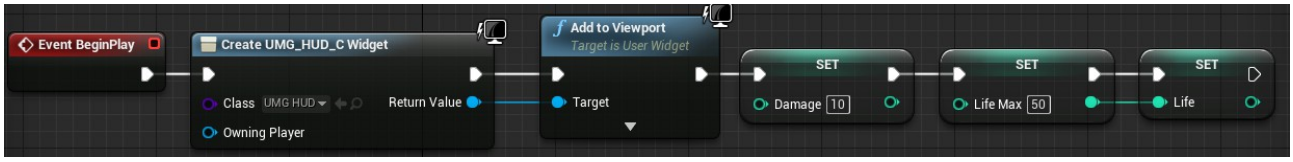
Commencez par créer **3 mapping d'axes** dans les **Inputs**. Il vous faut **Forward** mappé sur **Z** et **S**, **Right** mappé **D** et **Q**, **Turn** mappé sur **MouseX**.

Nous allons à présent créer les **déplacements** de base du joueur dans **l'event graph**.



On récupère l'appuie sur les touches et on fait bouger le personnage dans la bonne direction par rapport à son axe de devant. Pour tourner, on fait tourner le personnage grâce à la sourie.

Ensuite on va ajouter les **événements** de **départ**. Nous créons le **widget UMG_HUD** et nous **l'ajoutons** au **viewport**. Ensuite vous pouvez setter les **3 variables**.

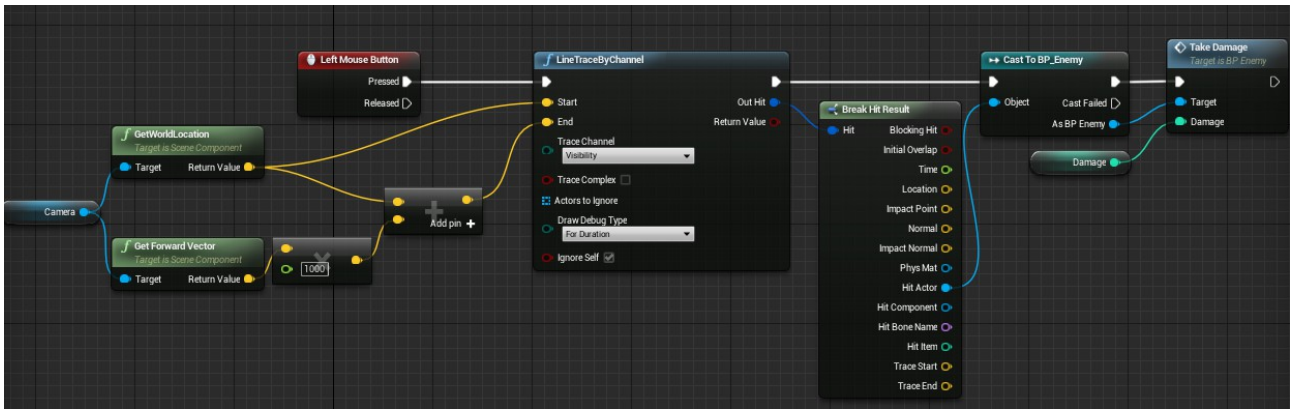


Pour le **tir**, nous allons utiliser le **linetrace**.

C'est un rayon que l'on envoi d'une **position** à une **autre** et qui **intercepte** tout ce qu'il **collisionne**. A condition que votre objet soit **bien réglé**, il **collisionnera** le **linetrace**. Une fois votre linetrace lancé, vous pouvez **récupérer l'objet** qu'il a rencontré, s'il y en a un.

On **cast** ensuite **l'objet** pour savoir si c'est l'objet que l'on **cherche** et on applique les dommages.

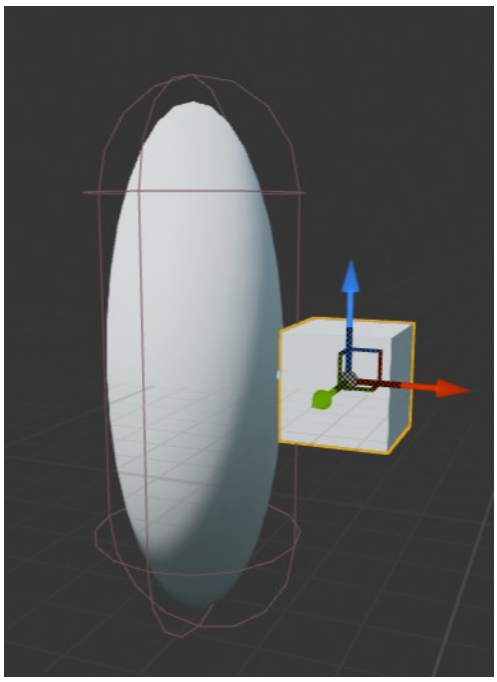
J'ai rajouté une **caméra derrière** le joueur en **décalée** pour avoir une vue de derrière en 3e personne. Vous pouvez aussi rajouter un **mesh** ou une **sphère** pour voir votre player (c'est plus **esthétique**).



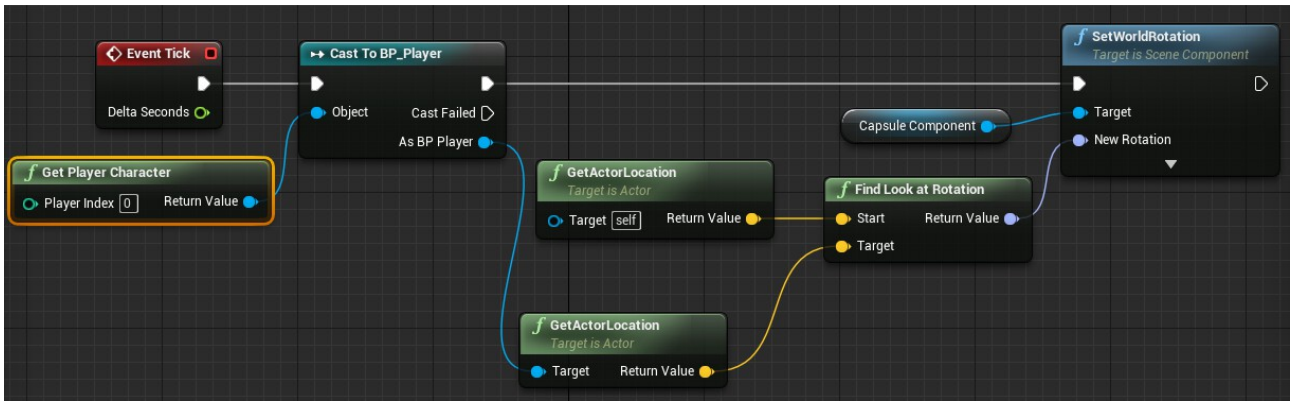
Avant de passer à l'ennemi, n'oubliez pas de **changer** la **Collison** de votre **capsuleComponent** à **BlockAllDynamic**. C'est ce paramètre qui permettra de **recevoir** les **linetraces**.

<u>Passons maintenant à **l'ennemi** :</u>

J'ai rajouté une **sphère** pour qu'on puisse le **voir** et j'ai également rajouté un **cube** devant lui afin de faire **partir** les **tirs** de cet endroit et de **voir** où se situe **l'avant** du personnage.
J'ai **nommé** ce cube **Gun**.



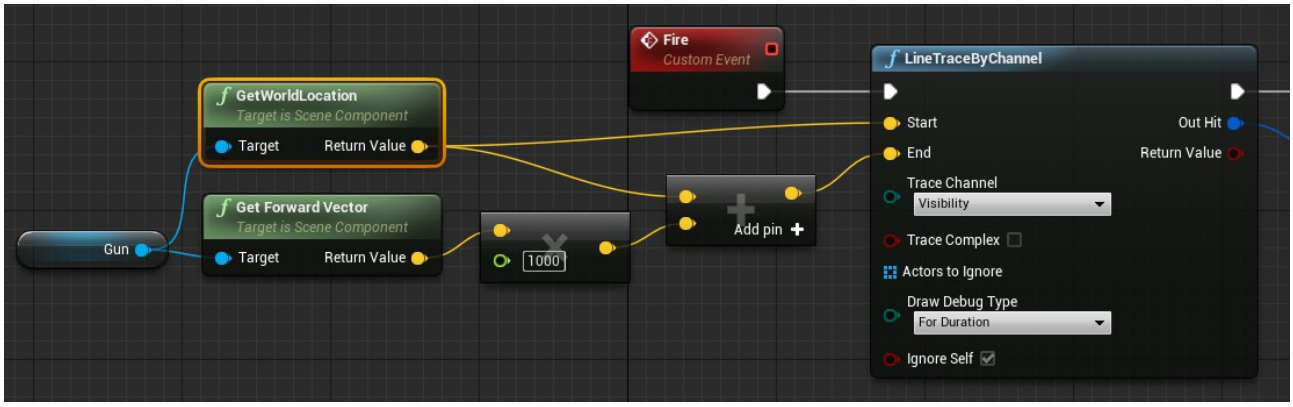
On va commencer par lui donner un **comportement** de **base** en le faisant **s'orienter face** au **joueur** en **permanence**.
Pour cela on va utiliser l'**Event Tick**. On **récupère** la **localisation** du joueur dans le **monde** et on recherche la **bonne rotation** grâce à la branche **Find Look at Rotation**. On **l'affecte** ensuite à la rotation de l'ennemi.

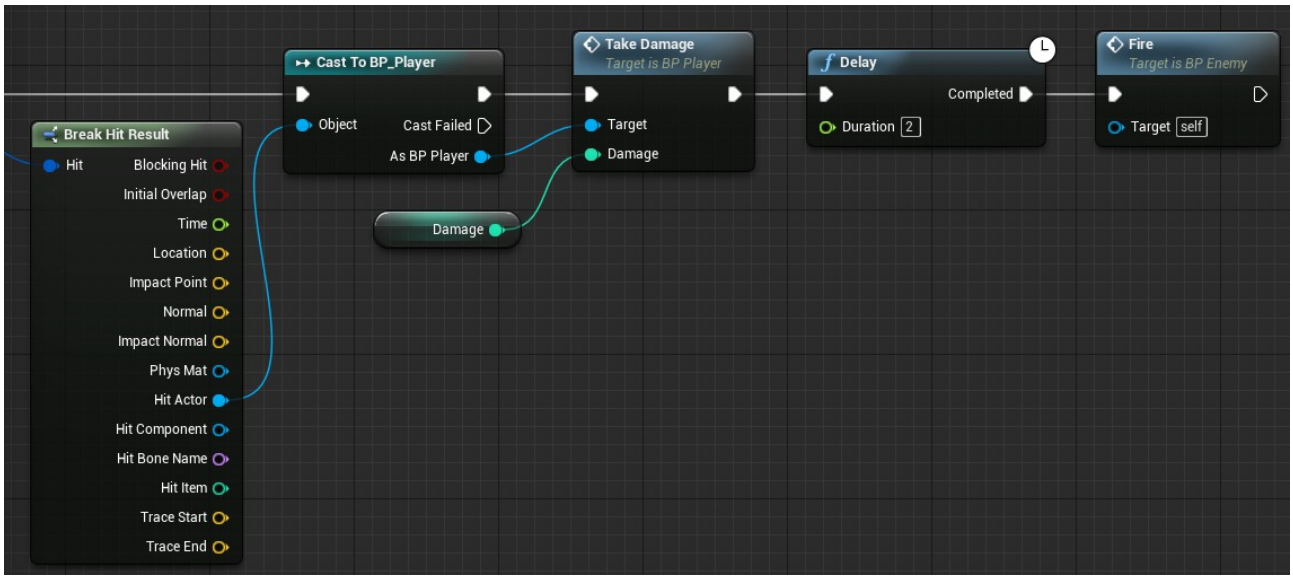


Ensuite on va le faire **tirer**. Créez un **Custom Event** et nommez le **Fire**. Cette fonction va fonctionner comme celle du joueur à deux/trois exceptions près.

On va faire **partir** les tirs du cube **Gun** et ils vont partir tout droit. **L'avantage** d'utiliser des **linetraces** c'est que le **tir** est **instantané** et qu'il n'utilise **pas** la **physique** pour calculer le touché sur une cible. On va lancer cette **méthode** en **boucle récursive**.

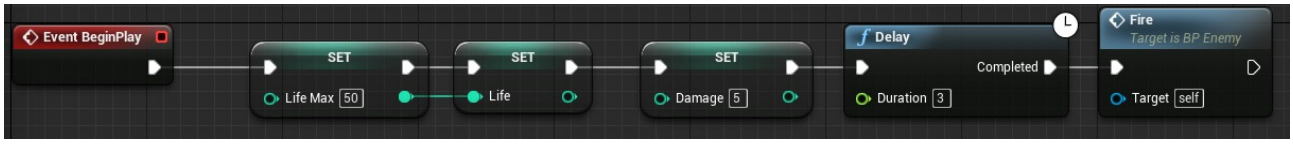
Je vous la met en deux partie. Essayez avant de faire par vous même à partir des explications, de votre compréhension et de votre souvenir de celui du joueur ;)





Ici on tire toutes les deux secondes. Vous pouvez changer le temps pour voir comment cela réagi.

Et enfin on affecte, au début du jeu, les variables et on lance le premier tir.



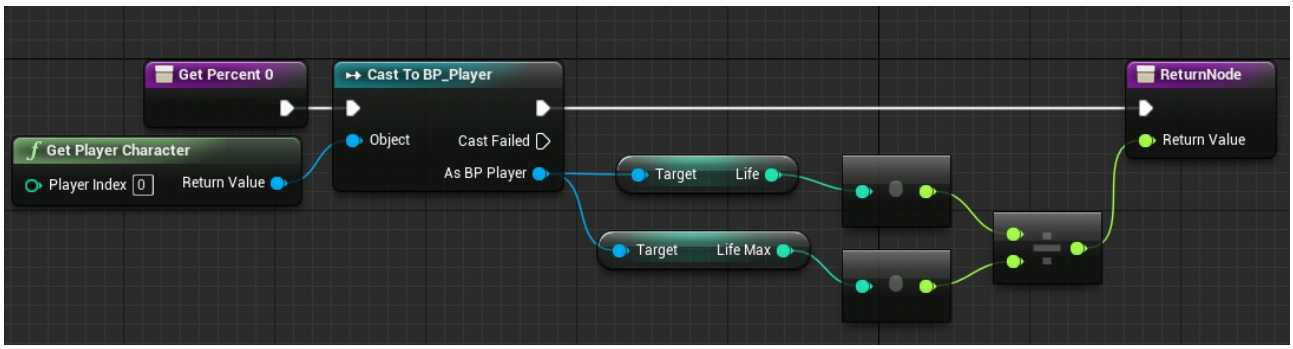
N'oubliez pas, comme pour le joueur, de mettre la **capsule component** en **BlockAllDynamic**.

On finira par un retour sur **UMG_HUD** avec le **bind** de la **vie** (et oui faut pas l'oublier!).

On sélectionne la **jauge de vie**, repérez **Percent** et faites un **Create Bind**.

On va simplement **diviser** la **vie actuelle** du joueur par sa **vie maximum**. Ainsi, le **ratio** variera entre **0** et **1**. Si vous souhaitez changer les **valeurs**, cela ne posera **aucun soucis**.

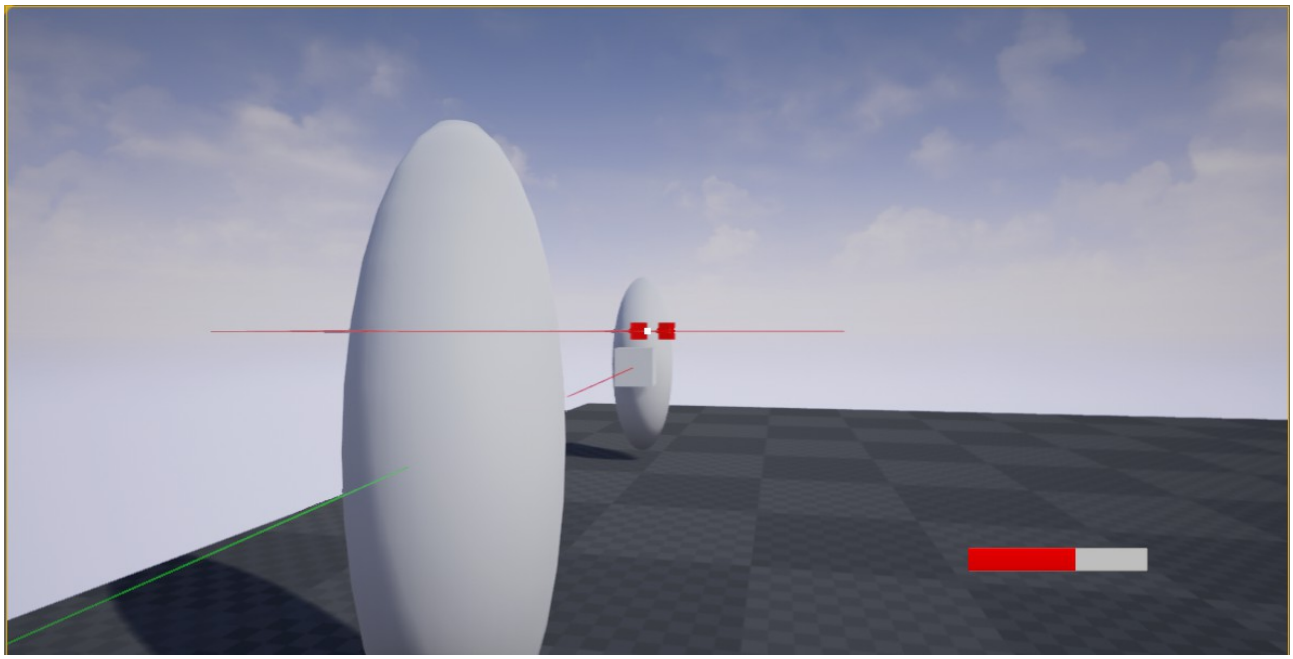
Le seul **problème** est que nos valeurs sont des **entiers** et que 45/50 en **entier** ça vaut **0**.
Pour cela on va **convertir** en **float** avant d'effectuer la **division** :



Il vous reste à **placer** un **ennemi** dans la scène et à **créer** un nouveau **GameMode**. Je l'ai nommé **BP_GameMode**. Changez le **Default Pawn Class** avec votre **BP_Player**.

Pour finir, **changez** le **GameMode** de base dans les **Settings** de **Maps & Modes** (**Edit**, **Project Settings**) par le vôtre.

Maintenant **lancez** le **jeu**, si vous attendez, vous verrez que vous **perdez** de la **vie** et si vous tirez sur l'ennemi, il **mourra** au bout de **5 tirs**. Si vous avez mis les mêmes options que moi, vous devriez **voir** les **traits** du **linetrace**.



Voilà qui conclu ce petit tuto sur les dégâts.

*Si vous souhaitez ne **plus voir** les **traits**, il suffit de **changer** le **Draw Debug Type** de **For Duration** en **None**.*

Pour toute éventuelles questions, remarques, détails ou demandes, n'hésitez pas à prendre contact avec moi ! (Cf : début du tuto)

Antoine Gargasson Septembre 2015