I- Interface/Viewport

A savoir absolument :

- **Quand on choisit le mode de jeu du projet, on peut toujours implémenter d'autres mode de jeu par la suite comme le First Person ou le Flying**
- **Si on supprime des fichiers dans le content browser, ceux-ci sont également supprimés de manière physique sur votre disque dur (ou SDD)**
- **Prévoyez toujours un backup de vos assets importés, en cas de modification, il sera plus facile de charger l'original.**
- **Pour compiler à la volée, cliquez sur Launch**
- **Afin de vous déplacer plus facilement sur le viewport, il est possible de créer des caméras fictives à travers le niveau et switcher entre elles, pour les créer : Ctrl+1,Ctrl+2 ; pour switcher : 1,2…**
- **sur le viewport : Ctrl+1, Ctrl+2...puis touche 1,2 pour switcher**
- **Pour voir les collisions directement sur le viewport (Ctrl+C)**
- **Pour repositionner plus simplement un actor ou une caméra, vous pouvez switcher entre le World Rotation et le Local Rotation (côté gauche de la grille)**
- **Le sens des flèches du Gyzmo indique le sens positif/négatif de l'objet en question**

Astuces :

- **Sur le viewport, nous avons la possibilité de faire un Screenshot en haute-résolution (pratique pour illustrer notre jeu lors d'une game jam ou vente)**
- **Dans l'onglet Windows, vous pouvez remettre la configuration des fenêtres par défaut en cliquant sur Default Layout**
- **Si vous rencontrez des soucis de performance, vous pouvez cliquer sur la flèche la plus à gauche du viewport et allez dans Screen Percentage, cela vous permettra de réduire la fenêtre du jeu**
- **Pour poser plus facilement un objet, pensez à la vue topdown**
- **Pour voir plus facilement la scène, vous pouvez retirer les lumières avec unlit**
- **Lighting only permet de voir comment la lumière réagit à votre environnement**
- **Lorsque vous faites un clic droit dans un espace vide et que vous choisissez une fonction, si celle-ci est accompagnée d'une parenthèse, cela signifie qu'un node supplémentaire va vous être présenté.**

II- Actors

A savoir absolument :

- **Un static mesh est un mesh n'ayant aucune possibilité d'animation d'où ce nom de « static »**
- **BP Sky sphere : Simule le ciel, les nuages etc…**
- **Directionnal light : Simule le soleil, dans la majorité des cas, il n'en faudra qu'un dans votre niveau**
- **Geometry : Formes géométriques destinées au blockout, possibilité de faire ctrl+C sur les arrêtes afin de modifier de manière avancée les formes**
- **Les collisions :**
  - **On peut modifier une collision grâce au Gizmo**
  - **On peut choisir des collisions pré-faites et les retoucher**
  - **On peut le faire soi-même manuellement**
- **Les lumières :**
  - **Static : Les lumières sont pré-calculées, pas de cast shadow**
  - **Stationnary : Pareil que le précèdent mais avec cast shadow**
  - **Movable : Tout en temps réel**

I- Actors

A savoir absolument :

- **Toujours considérer les blueprints actors comme des classes (poo)**
- **Chaque type de Blueprint possède des components de bases (l'équivalent du starter content)**
- **DefaultSceneRoot : Visualisation 3D de base**
- **Bien se rappeler que chaque blueprint ou instance se comporteront de la même façon**
- **Pour toute action commune entre 2 objets, étudiez la possibilité d'héritage de classe**
- **Sur la fenêtre située à gauche du viewport blueprint, il est possible d'associer/dissocier des components**
- **Les lignes d'un tableau sous UE se nomment « elements »**

Astuces :

- **On peut ajouter plusieurs collisions dans son blueprint, ce qui peut servir à titre d'exemple à localiser les dégâts dans son jeu**
- **Bien comprendre qu'un bp arme doit présenter toutes les caractéristiques d'une arme (tirer, recharger, munitions...), nul besoin de recréer un bp pour un ak-47 ou un revolver, choisissez un système d'héritage.**
- **On peut ajouter un begin overlap ou autre à partir de la collision du blueprint**
- **N'oubliez pas que l'on peut inclure plusieurs mesh dans un blueprint !**
- **Dans begin overlap, faire un cast playercharacter et joindre le target à other actor**

II- Components de Mouvements

A savoir absolument :

- **Pour actionner la rotation d'un actor, il faut mettre le defaultsceneroot sur movable**
- **Si vous cochez « Component activate », l'objet va tourner par défaut**
- **Pour gérer un déplacement -> Interp to movement, allez dans details et voir Duration et control points**
- **Lorsque l'on veut simuler que plusieurs objets bougent à la chaîne, on peut utiliser un loop reset (en ayant pris soin de mettre 2 entrées) pour créer l'illusion**
- **Lorsque position is relative est cochée, nous n'avons pas besoin d'indiquer la location de départ d'un objet**
- **Le behavior permet d'intéragir sur la fin de notre mouvement**

III- Custom Events/Fonctions

A savoir absolument :

- **EventBeginplay : Comprendre : « Quand l'actor apparaît dans le niveau, exécuter ceci… »**
- **Les events sont comme des fonctions simplifiés que l'on appelle directement sur l'event graph**
- **Dans une fonction, ce qui est inscrit est obligatoire et ce qui est dépliable est optionnel !**
- **Le start time d'un sound signifie que celui-ci zappera ce qui précède la seconde définie, autrement dit, si l'on met 2 secondes, les 2 premières secondes du son ne seront pas joués !**
- **Function override : Permet de voir les fonctions préfaits d'UE en fonction du blueprint en question**
- **Lorsque l'on met un actor en variable, on ne fait que pointer dessus, autrement dit le mettre en référence (ne consomme pas de ressources)**
- **Maintenir alt en droppant une variable pour la mettre directement en position SET**
- **L'œil ouvert (ou éditable en jeu) permet d'insérer une valeur unique à un blueprint et différencier les instances**
- **On peut ajouter des paramètres sous formes de pins sur une fonction**
- **On actor : On peut caster n'importe quel blueprint et le mettre en hit actor (break hit result) mais on ne pourra pas par ce biais différencier l'actor en question car il va s'agir de savoir si l'actor fait partie de la classe susnommée !**
- **On peut utiliser un get all actor of class si vous n'avez qu'une seule instance du bp_objet en question, en out actors, il faudra mettre get, une condition == que l'on relie à hit actor, un branch à all actor of class et après vous pouvez mettre la suite du code**
- **Pour différencier les instances, on peut utiliser un tag, puis faire un get all actor with tag**
- **Retriggerable Delay Vs delay = Si l'on associe un événement, les secondes du RDelay seront réinitialisés alors que dans le delay non**
- **Add ActorWorldOffset = Permet de téléporter des objetes**
- **Do N : Permet de faire x fois une action**
- **Do Once : On peut mettre un custom event pour reset le do once, si on clic sur closed, il faudra obligatoirement passer par le custom event pour reset**

IV- Flow Control

A savoir absolument :

- **For loop : Permet de faire une boucle**
- **For loop with break : Même chose mais on peut la stopper via un Custome Event**
- **Gate : Tout ce qui suit exit est exécuté si la gate est ouverte, le toggle sert à alterner selon la position en temps réel de la Gate**
- **Multigate : Permet d'ajouter plusieurs actions**
- **Sequence : Ce flow control n'attend pas que tous les différents nœuds soient terminés pour passer à l'autre, il n'y pas non plus de lien entre les différents nœuds**

A savoir absolument :

Pour faire bouger notre personnage, le spring arm permet deux choses :

- **Faire en sorte que lorsque le personnage est dos à un mur, la caméra ne traverse pas le décor**
- **Permet d'associer plus facilement des mouvements de caméra**

Il existe des codes couleurs pour chaque fichier dans le content browser :

- **Rouge pour les textures**
- **Vert pour les matériaux**
- **Vert foncé pour les animations…**

Lors de l'importation de personnages mixamo, on peut trouver certaines incompatibilités mais qui peuvent être réglé en modifiants les textures par exemple.

Pour créer un effet de transition, nous pouvons cocher le camera lag

Astuces :

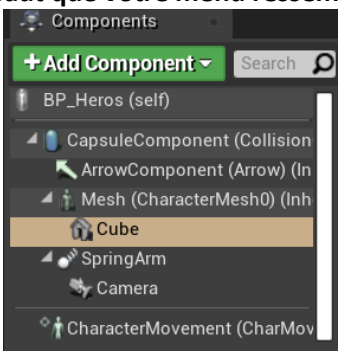
3 composants d'un personnage :

- **Le Mesh**
- **Le Skeleton (squelette de notre personnage)**
- **Le Physics (permet de localiser les différentes parties du corps, ex : localisation des dégâts)**

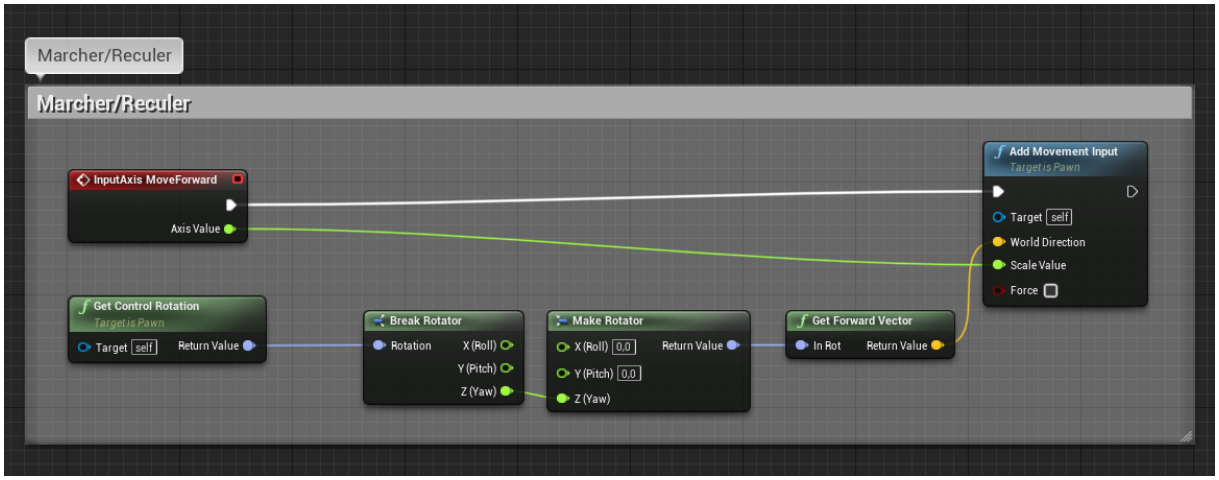
Sur le mesh, on peut surligner ou isoler ce qui nous intéresse (highlight,isolate)
On peut passer d'un mode à l'autre en haut à droite

Manipulations :

Ouvrir un projet blank sans starter content
Créer un bp_héros, une fois dedans, il faut que votre menu ressemble à ceci :

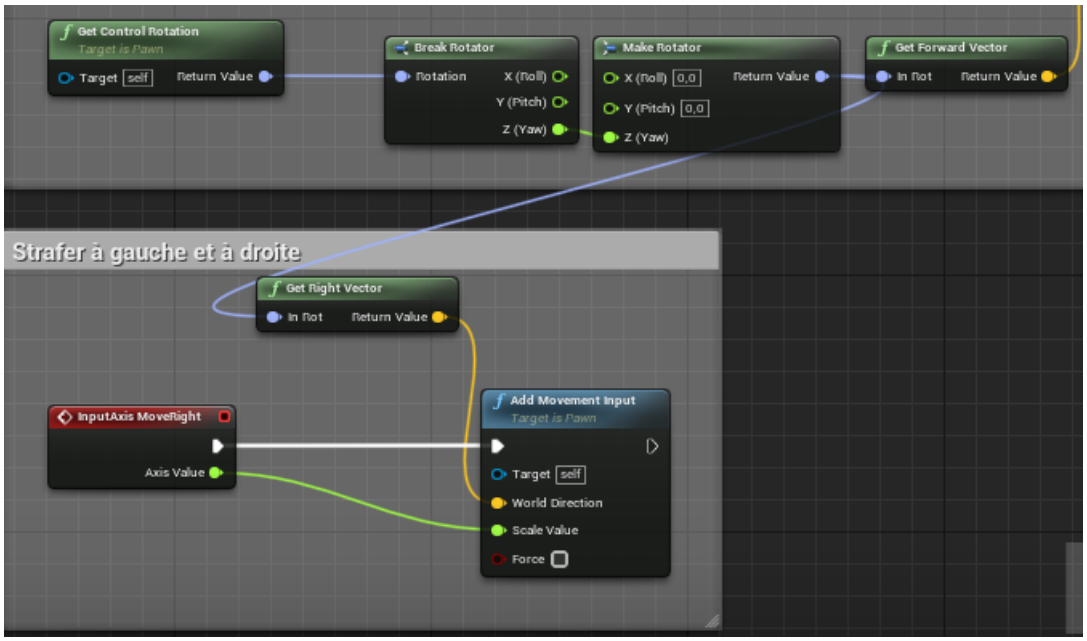


Allez dans input, mettre dans axis mapping « walkforward », mettre Z et S, Z avec un scale à 1 et S avec un scale à -1,
Créer un bp_perso avec le blueprint suivant :



L'inputaxis MoveForward correspond à notre touche X ou Z, le scale que l'on a mis correspond au scale value
Le Get Control rotation permet d'avoir les coordonnées du joueur et le make rotator permet de les modifier
Enfin le Get Forward Vector permet de récupérer la nouvelle valeur de la position du personnage.

Pour strafer de gauche à droite, reproduisez le schéma suivant, allez dans input et mettre les valeurs correspondantes de la même façon que vous avez fait précédemment (Q & D) :



Pour jouer avec la caméra de la souris, input, axis mapping, renommer « turn » choisir mouse puis x, pas besoin de mettre autre chose car UE calcule le négatif et le positif par rapport à la position de la souris
Dans le BP, mettre inputAxis Turn suivi d'un add controller yaw input
Dans le BP, mettre inputAxis Lookup suivi d'un add controller pitch input
Dans input, mettre -1 pour le lookup sinon la vue caméra sera inversée

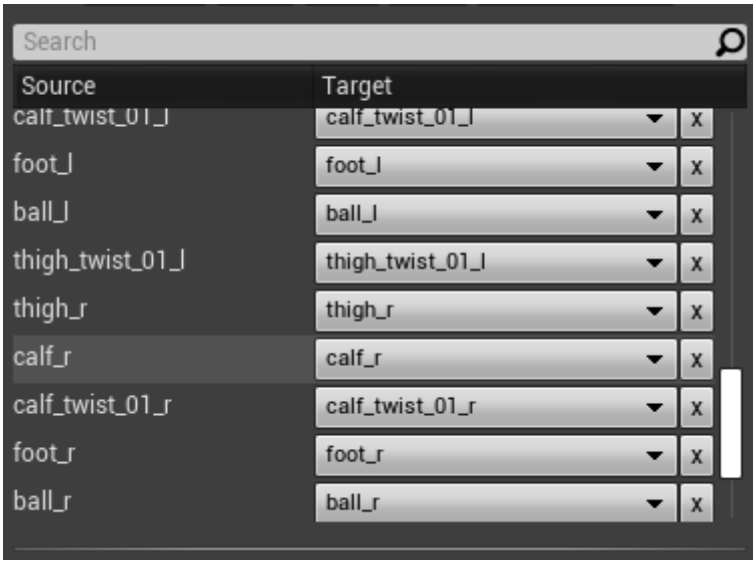
Important, dans le spring arm, cocher camera rotation sinon ça ne fonctionnera pas !!!
Pour sauter -> action mapping -> jump en pressed et stop jumping en released

Faire un jeu de combat TPS (Evan's Game Teacher)

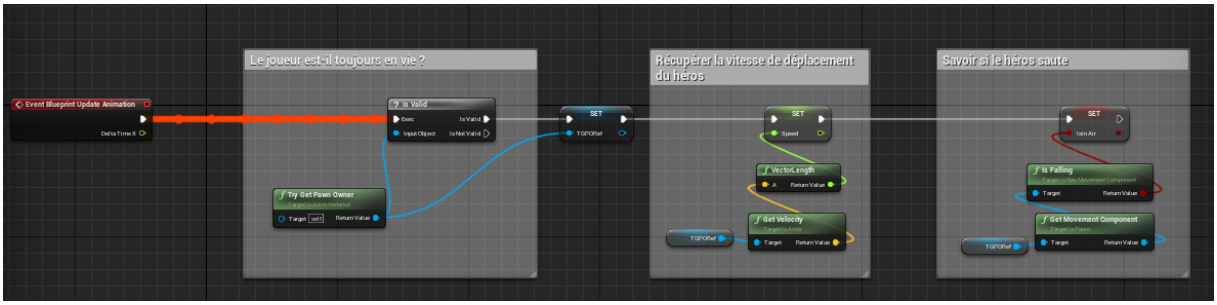
1/ Retarget Manager

Permet de faire en sorte que notre personnage de base puisse disposer du squelette d'un personnage plus avancé

Importer le personnage wukong
Allez sur le mannequin skeleton, choisissez retarget manager, choisissez onion, puis cliquez sur save
Attention : Regardez bien si tout s'assemble correctement avant !



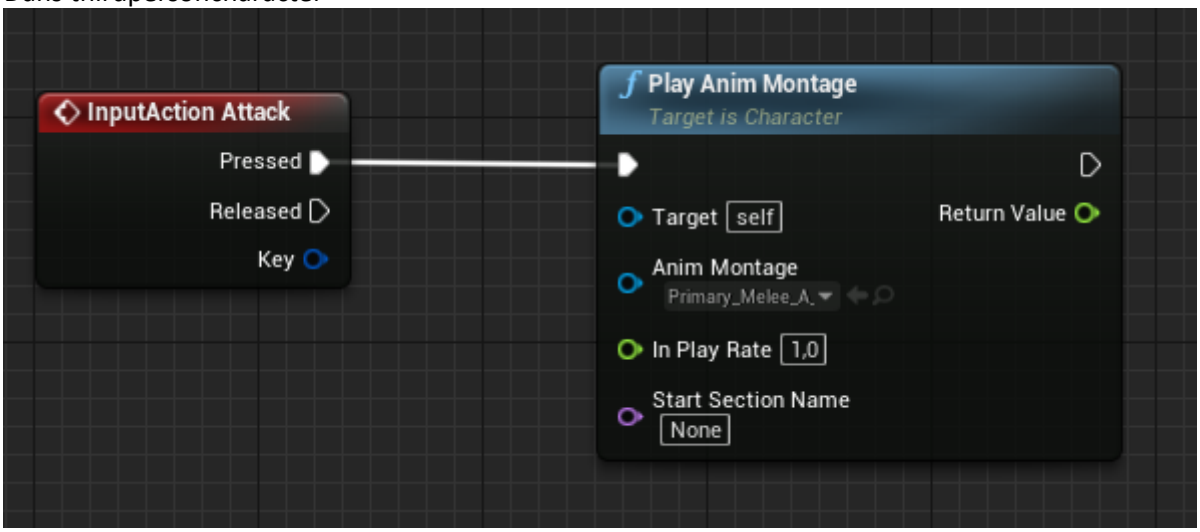
Allez dans les animations du paragon, choisissez idle, jog_fwd, jump Apex et Jump Land, clic droit « retarget anim assets », clic sur mannequin skeleton
Clic sur change, créer un nouveau dossier dans content appelé « animation »
De retour, clic sur retarget
Dans le rép animation, clic droit et mettre un animation_blueprint, choissiez le skeleton du mannequin et clic sur ok
Allez sur l'event graph et reproduisez ce shéma :



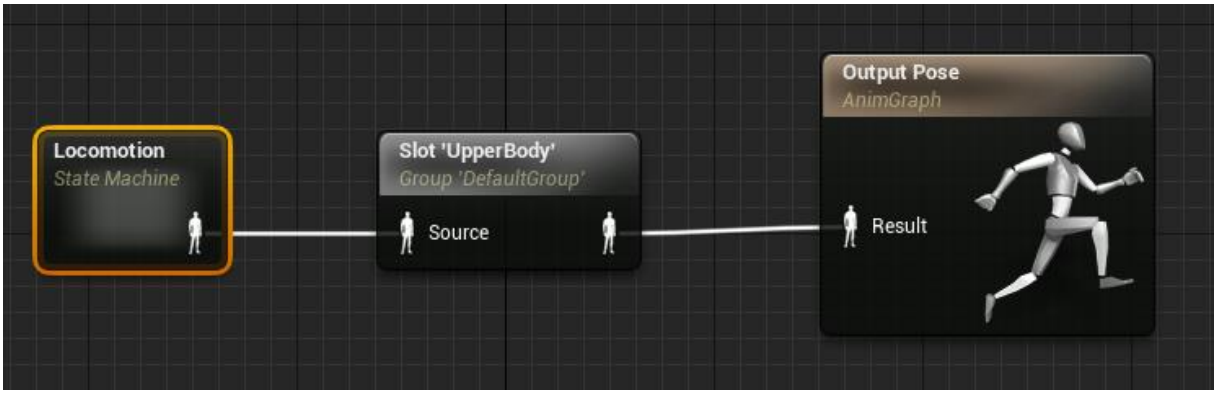
Dans l'animgraph, créer un new state machine « locomotion »
Double click sur ce dernier, add new state, « idle », double clic sur idle et drag le idle, reliez le tout.
Remontez d'un cran, relier le idle à l'entry, et save (vous devriez voir le personnage bouger)
Dans locomotion, partez de Idle, add new state et walk, ajouter le jo_fwd dans le walk comme précédemment
Double clic sur l'icone de transition et reproduiez le schéma suivant

Sur la flèche retour faire en sorte de vérifier si la personne est à l'arrêt
Vérifier que la personne saute dans le state jump
Vérifier que la personne est ne saute pas dans le state land
Vérifier que la personne atterit (cocher entrer transition)
Allez sur le bp_player, viewport, clic sur le mesh et mettre bpa_player dans animation
Allez dans le content browser, view options, cocher view engine content
Faire une recherche cylindre et le copier dans content
Créer un dossier props et mettre le cylindre, renommez-le en SM-Stick
Créer un matériaux M_stick
Clic dessus et mettre une couleur en mettant un constant3vector
Dans le skeleton tree, créer un socket sur le hand_r « WeaponSocket »
Mettre dans le relative scale les valeurs 0.04,0.05,2.0
Faites en sorte de mettre la bonne position en comparant avec le skeleton de wukong
Pour savoir si le baton reste bien accroché, allez dans preview animation
Dans le mesh du viewport de notre thirdpersoncharacter, ajouter le baton avec le socket et repositionnez-le bien, sauvegarder tout.

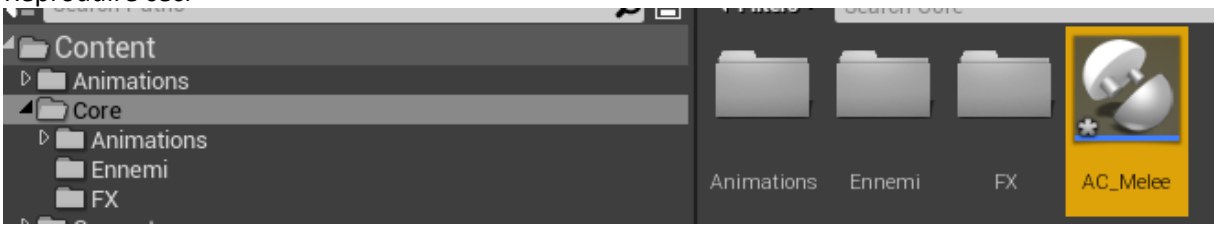
Faire des combos :
Allez dans les animations de wukong, prendre les primary a,b,c,d,e slow en vert et en bleu puis faire un retarget, change, newfolder « attacks », retarget
Facultatif : Choissiez un primary a montage et faites en sortes qu'ils appartiennent tous au upper body
Dans project settings, mettre le bouton attack
Dans thirdpersoncharacter



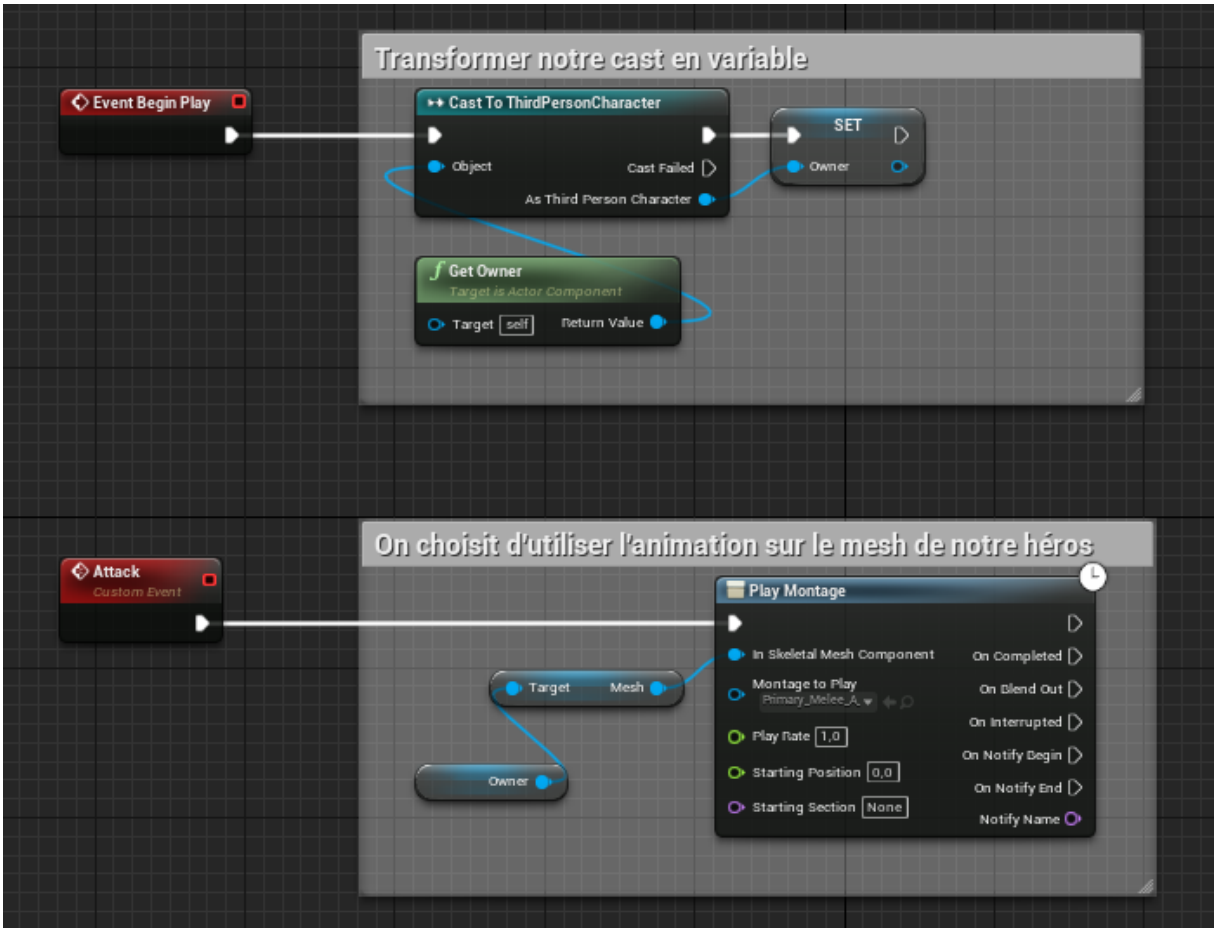
Reproduisez le schéma suivant :



Reproduire ceci



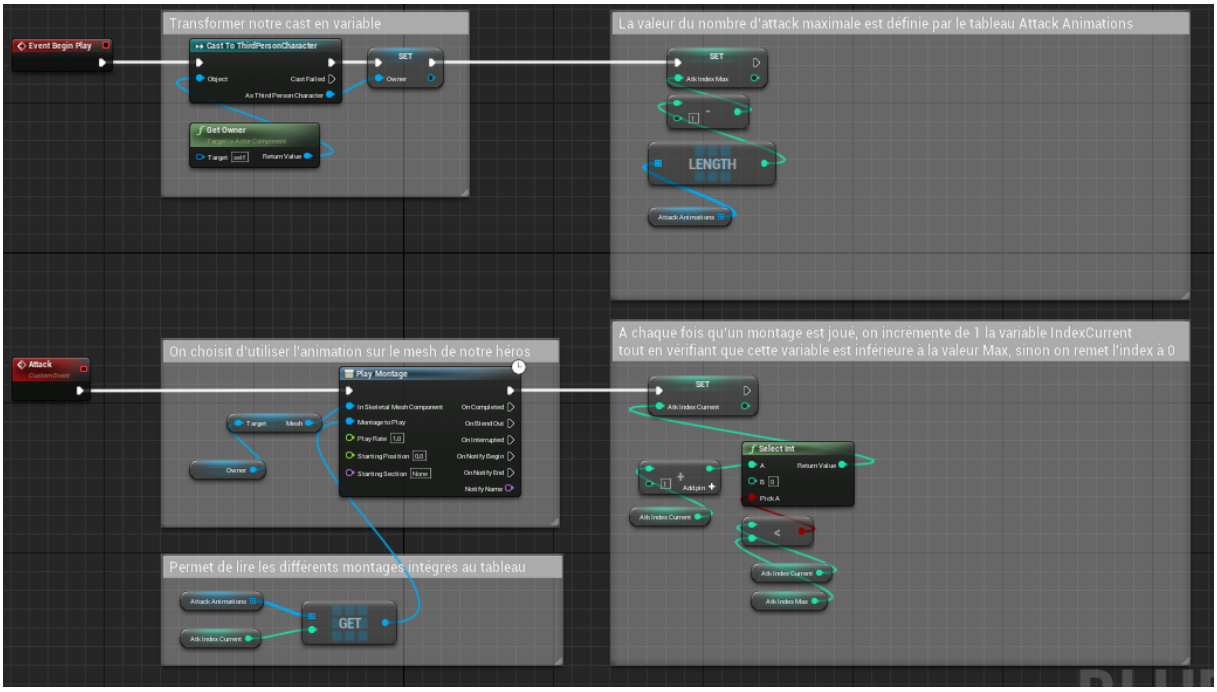
Dans thirdpersoncharacter, add componet « Ac_Melee »
A noter : On utilise un actor component pour l'évolutivité du jeu, on pourra changer plus facilement de personnage en utilisant ce procédé



Normalement, on rajoute des recovery sur le montage mais la flemme de réimporter tout le truc

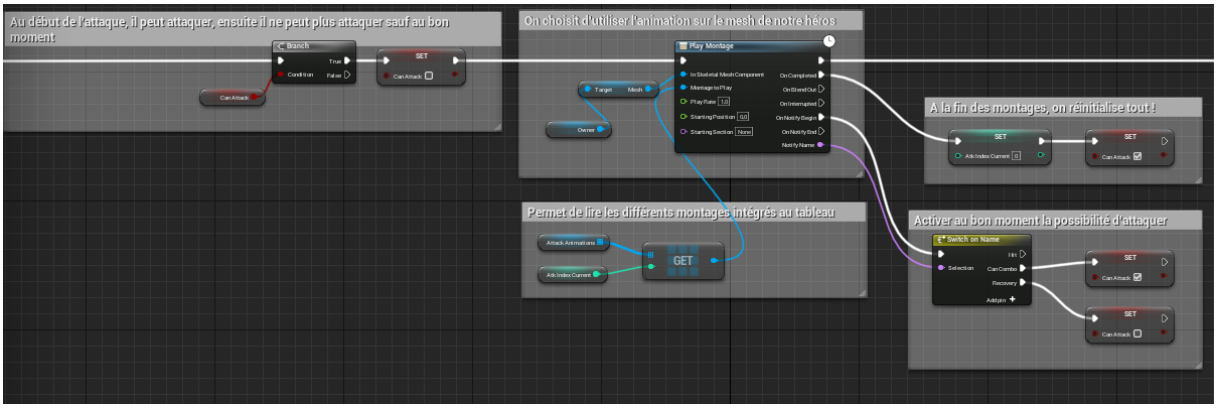
Dans AC-Melee, créer variable « AttackAnimations », type Anim montage et tableau et renseignez les différents animations montages. Là j'ai du fabriquer un animation montage pour le E car il n'existait pas.

Créer 2 integer « AtkIndexCurrent » et « AtkIndexMax »

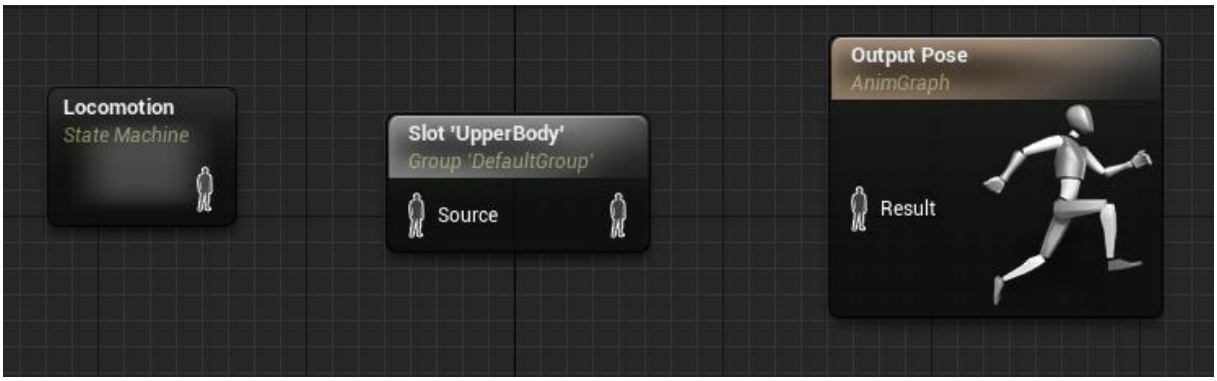


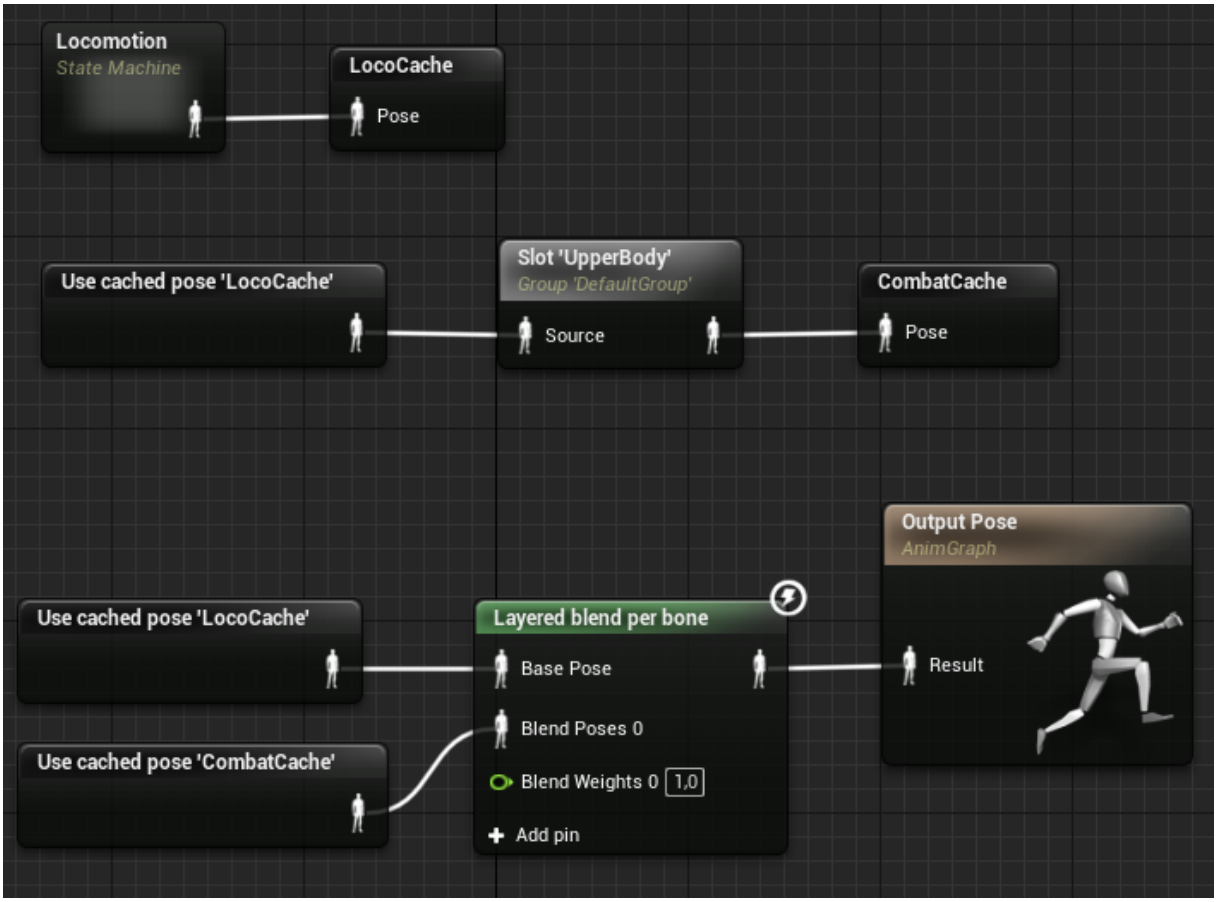
On peut mettre le blend à 0 si on le veut
Dans le dossier Core : Créer un blueprint anim notifiy « MN hit »
Double clic dessus -> name « hit », couleur jaune, compile
Dupliquez deux fois le MN HIT

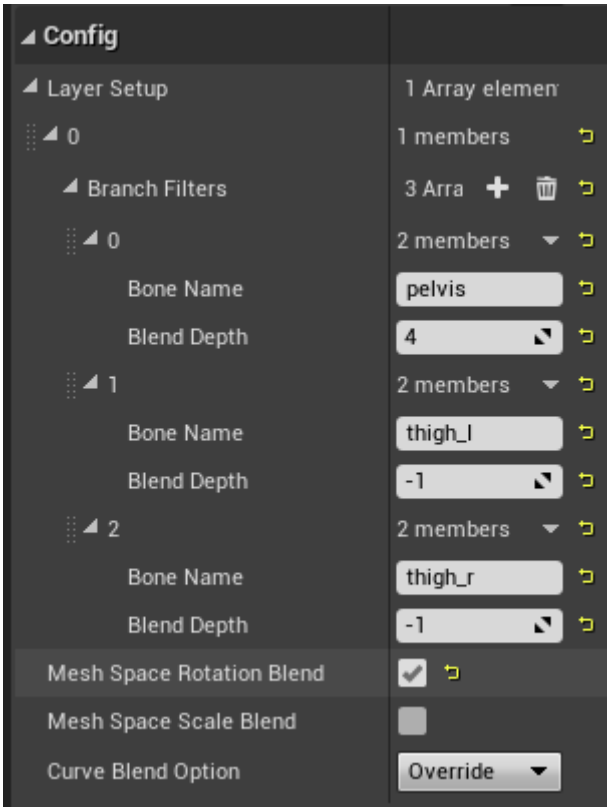
Dans les différents montages, placé les 3 aux bons endroits
Hit à l'endroit où le héros est censé taper
CanCombo à l'endroit où le héros est censé pouvoir faire un combo
Recovery à l'endroit où le héros ne peut plus faire un combo



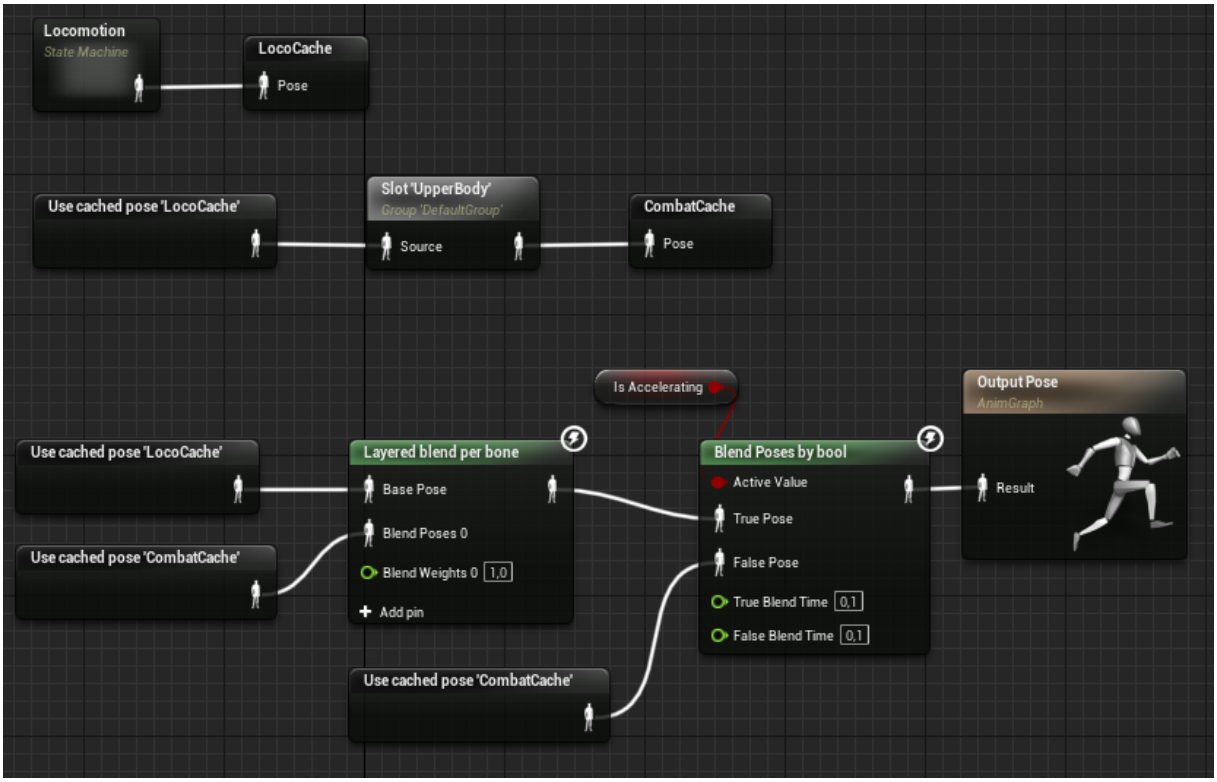
Test





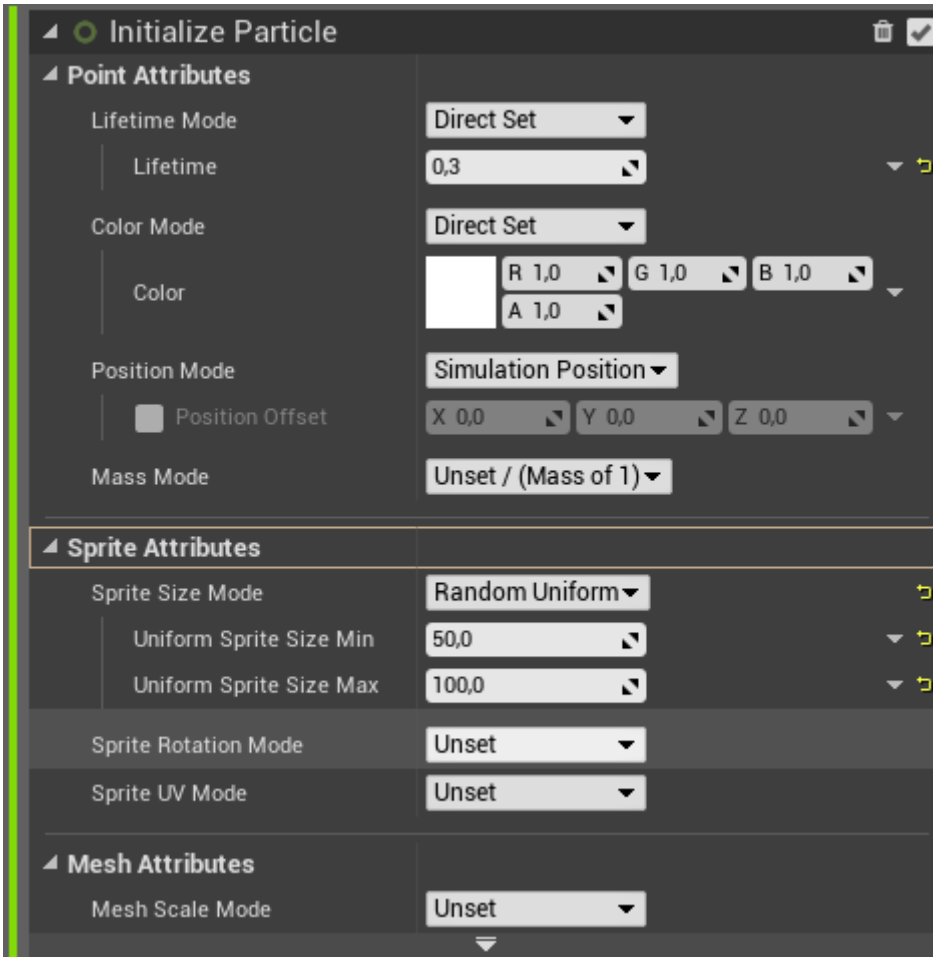


Facultatif : faire une attaque jump (voir vidéo si besoin)

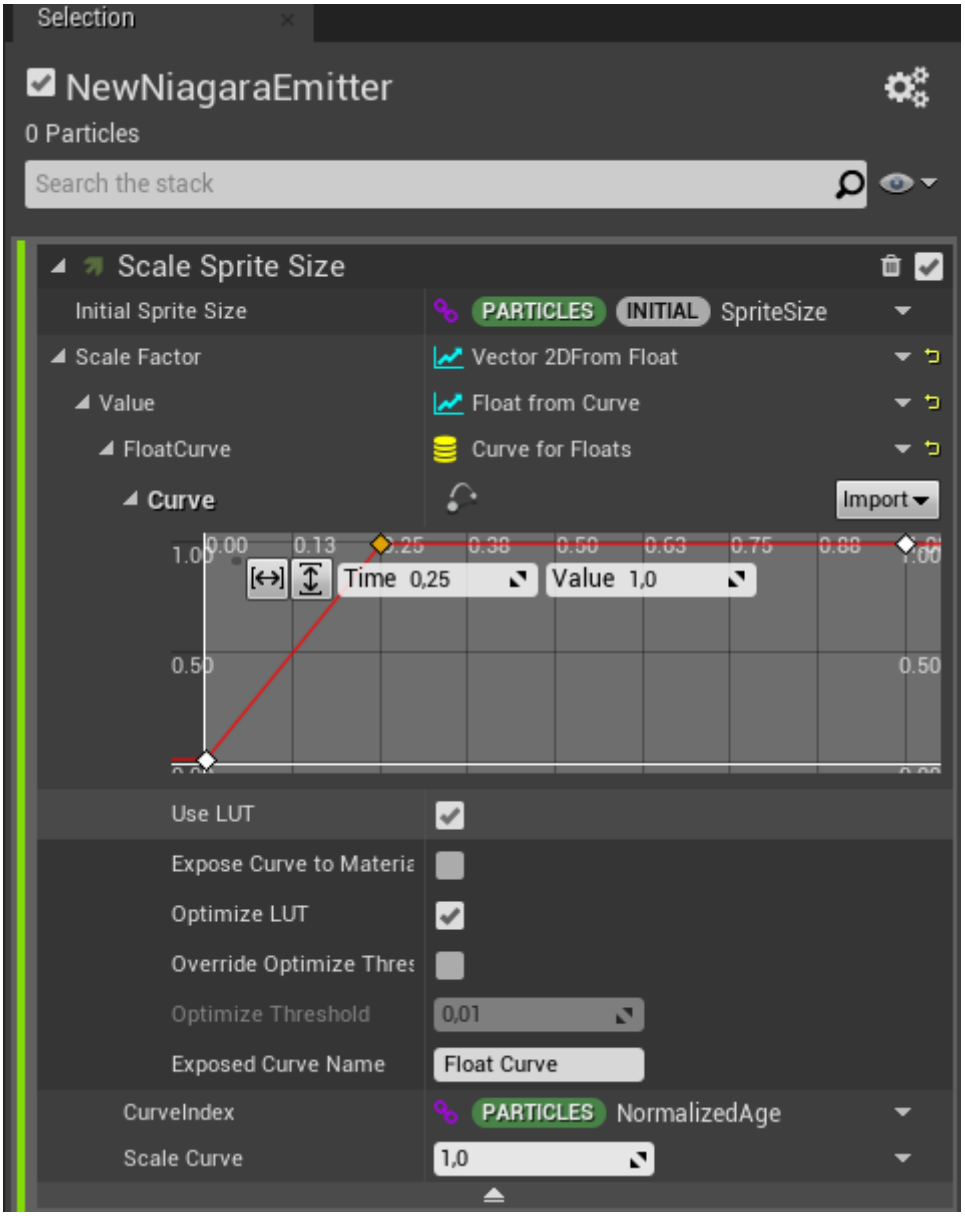


Les VFX :

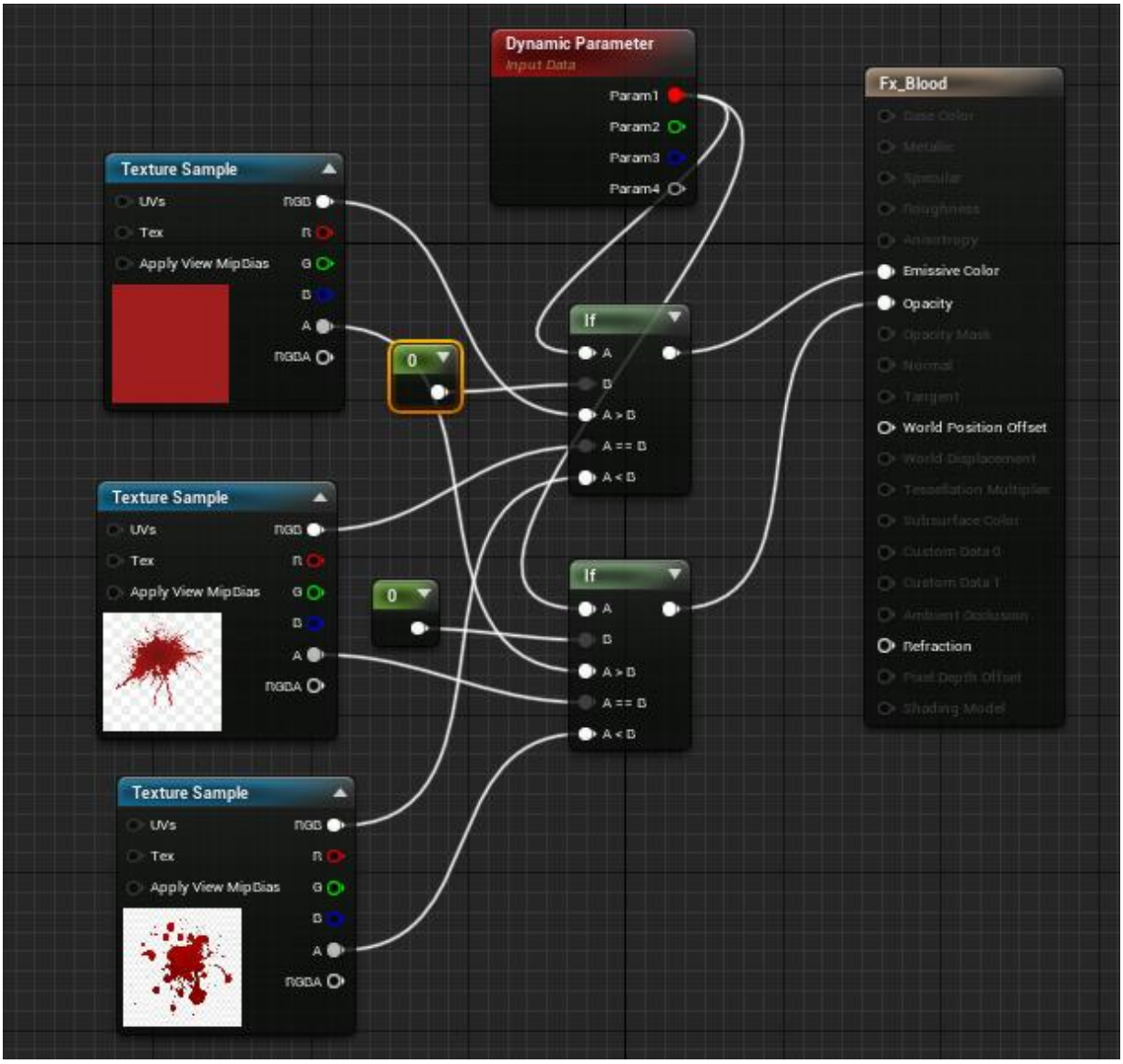
Core->fx->niagara emittter->new emitter->simple sprite burst->
Emitter state à 1



Particle spawn + sphere location -> radius 200
Particle update + scale sprit size-> Vector 2D from float

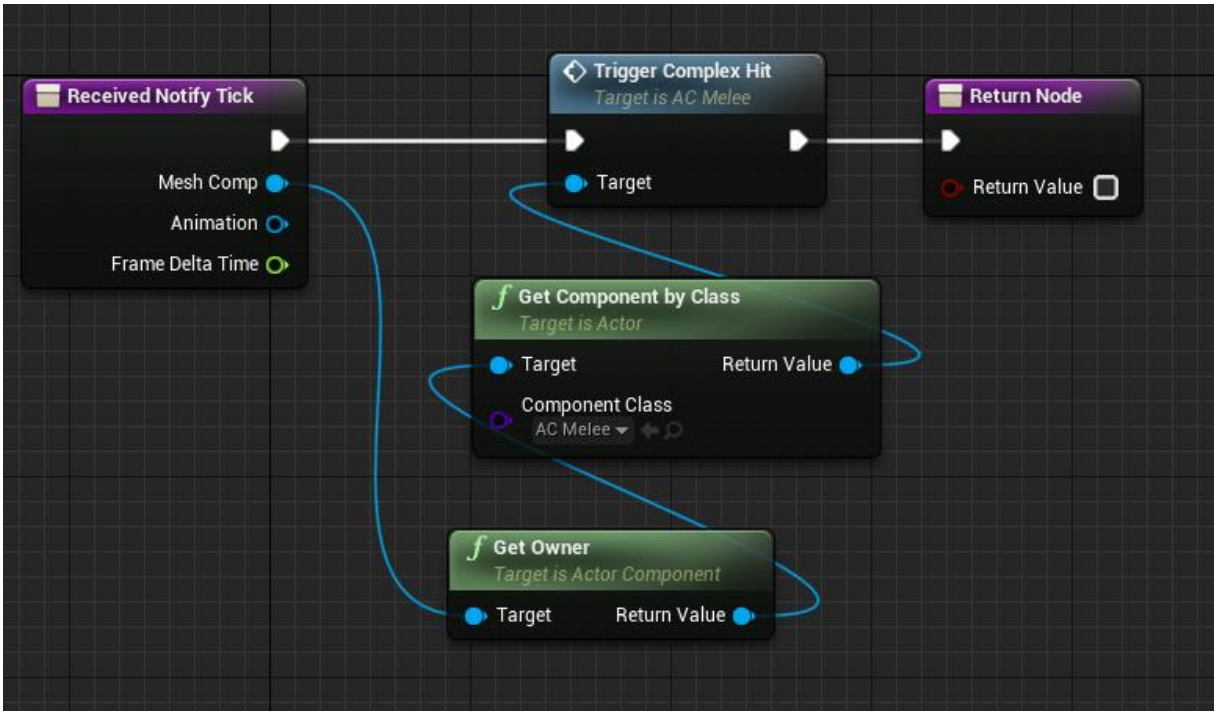

Selection
NewNiagaraEmitter
0 Particles
Search the stack
Scale Sprite Size
Initial Sprite Size
PARTICLES
INITIAL
SpriteSize
Scale Factor
Vector 2DFrom Float
Value
Float from Curve
FloatCurve
Curve for Floats
Curve
Import
0.00
0.13
0.25
0.38
0.50
0.63
0.75
0.88
1.00
Time 0,25
Value 1,0
0.50
Use LUT
Expose Curve to Materia
Optimize LUT
Override Optimize Thres
Optimize Threshold
0,01
Exposed Curve Name
Float Curve
CurveIndex
PARTICLES
NormalizedAge
Scale Curve
1,0

Test

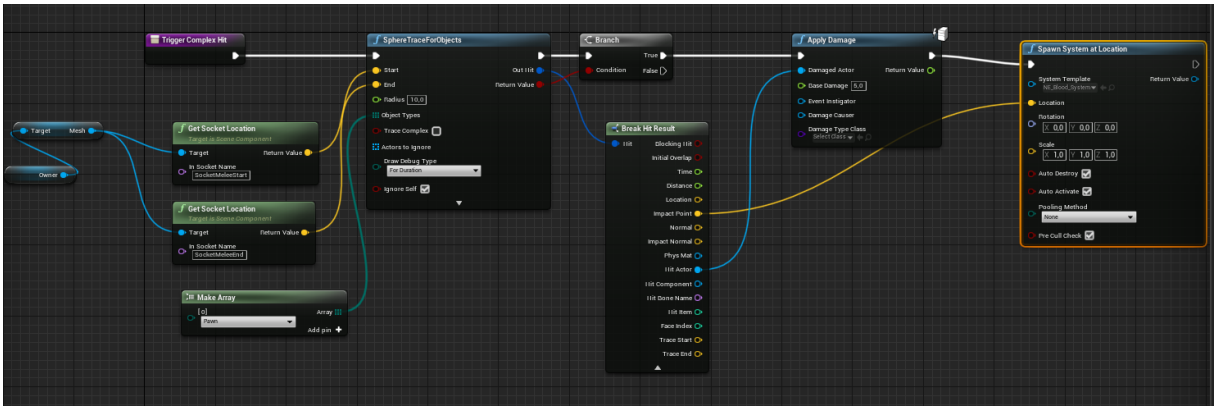


Combat réaliste

Créer une fonction « TriggerComplexHit » dans ac_melee
Créer dans core un anim notify state « AS_DealDamage »
Ajouter le As_DealDamage un peu avant le MN_hit pour finir à la fin du MN hit dans l'animation
Le faire pour toutes les animations
Allez dans le //, override, received notify tick



Dans le skelette du héros, mettre 2 sockets au hand_r « SocketMeleeStart » et « SocketMeleeEnd »



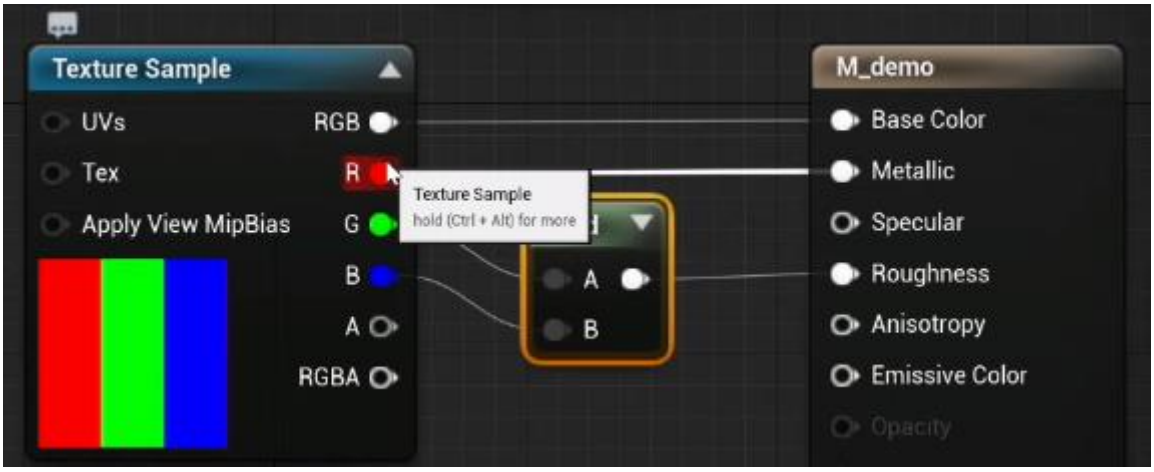
Les matériaux

- **PBR :** Se dit d'un matériau qui vise à réagir de façon réaliste avec son environnement
    - 3 composants obligatoires pour le PBR : Specular, Metallic, Roughness
    - Les valeurs sont comprises entre 0 et 1
    - On utilise une constante pour définir les valeurs

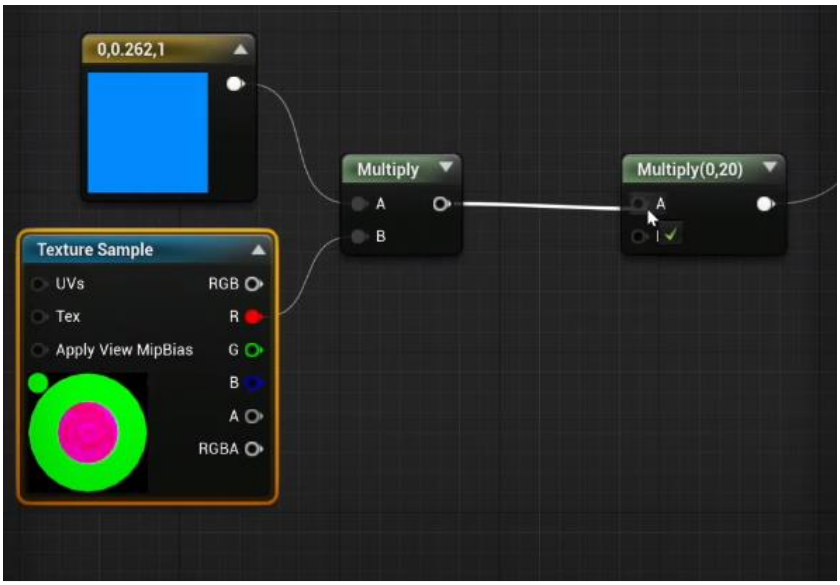
Ex : donner un effet plastique : Metallic 0, Roughness 0.5

- Le base color attend toujours les couleurs rouge,vert,bleus
- Pour changer le base color, on utilise un constat3Vector

« Cleanup » permet de retirer tout les nodes non utilisés !
Texture Group : Permet de regrouper les textures et ainsi optimiser plus facilement le jeu
Le roughness apporte des nuances de gris
Pour ajouter du metallic à une couleur, on prend le texture sample, on ajoute le rgb à la base color et on lie le R à Metallic
Pour lier plusieurs couleur à un aspect, on utilise add (cf image)



Lorsque l'on souhaite choisir qu'une partie de son mesh (ex : spot lumineux) pour le rendre brillant, il faut créer une image en noir et blanc, tout le noir sera ignoré et le blanc sera appliqué, il faudra s'assurer que l'image se scale bien avec le mesh pour avoir un bon rendu.
On peut également se servir de la couleur rouge, vert ou bleu pour faire la même chose.



- La normalmap est ce qui donne l'impression de relief
    - La normalmap permet également d'avoir un rendu high poly à partir d'un modèle low poly
    - On travaille toujours et uniquement sur le rgb avec les normalmap (que l'on relie à normal)
    - On ne travaille pas sur une normal map sur photoshop mais sur des softs spécialisés !

Les masters materials (ou Materials instances) permettent de regrouper plusieurs types de textures qui se ressemblent (bois,metal…) ou qui sont associés à un environnement dédié (intérieur, extérieur). Lorsque le master materials aura été modifié, toutes les textures appartenant à ce master(groupe) hériteront de la modification.

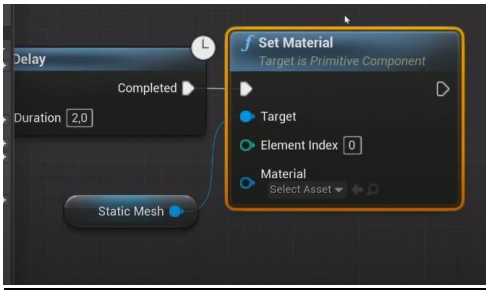
Créer un m_master, puis pour créer une instance, clic droit dessus et créer « material instance » L'objectif est par exemple de garder l'aspect métallique et de juste changer la couleur, si l'on change les propriétés du master matérial, ex : roughness 1, tout les instances vont changer en ne reflétant plus la lumière.

Les materials intances parameters

Dans le master material, Clic droit sur une constante et « convert to parameters », ce qui permettra aux instances de pouvoir paramétrer par exemple le roughness ou le metallic.
Nous pouvons le faire pour tout, ex : texture, ce qui permettra aux instances de pouvoir paramétrer la texture

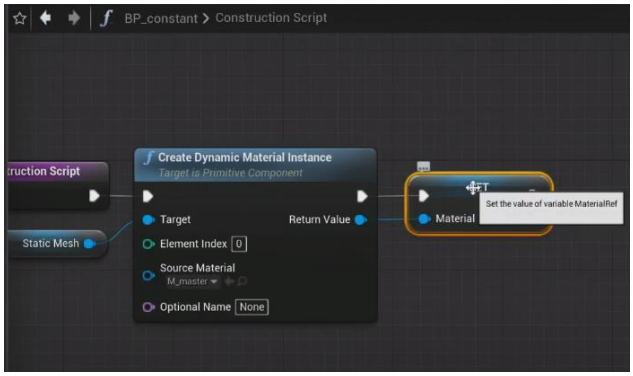
Changer de material en cours de jeu

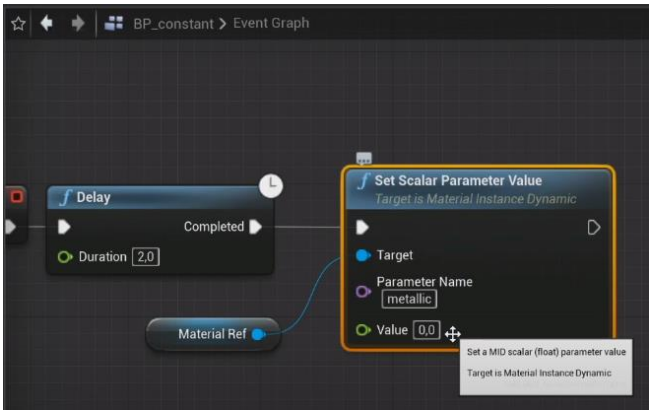
Event begin play puis



Dynamic Material Instance

Prenons l'exemple du jeu de voiture, dans celui-ci on peut choisir sa couleur, y'a 2 façons de procéder :
1- On crée autant de « material instance » qu'il y a de couleurs que l'on souhaite dans le jeu (faible en ressources)
2- On donne au joueur la possibilité de choisir lui-même la couleur par un « dynamic Material Instance » (Moins lourd en travail fourni, pas besoin d'instance mais plus lourd en mémoire)

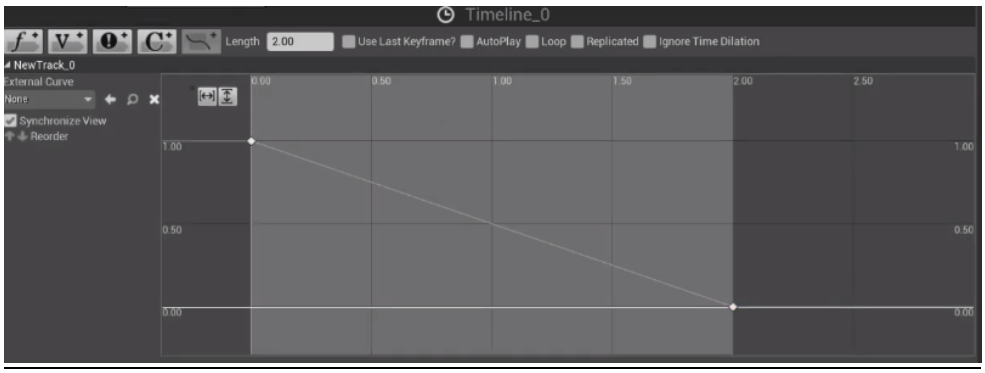




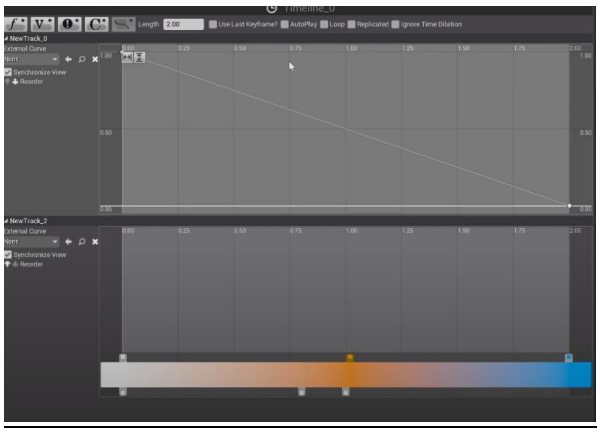
Donc sans créer d'instance et sans changer de texture, on a donné la possibilité de modifier une valeur en temps réel d'un paramètre souhaité, ce qui va beaucoup plus loin que de simplement changer la texture.

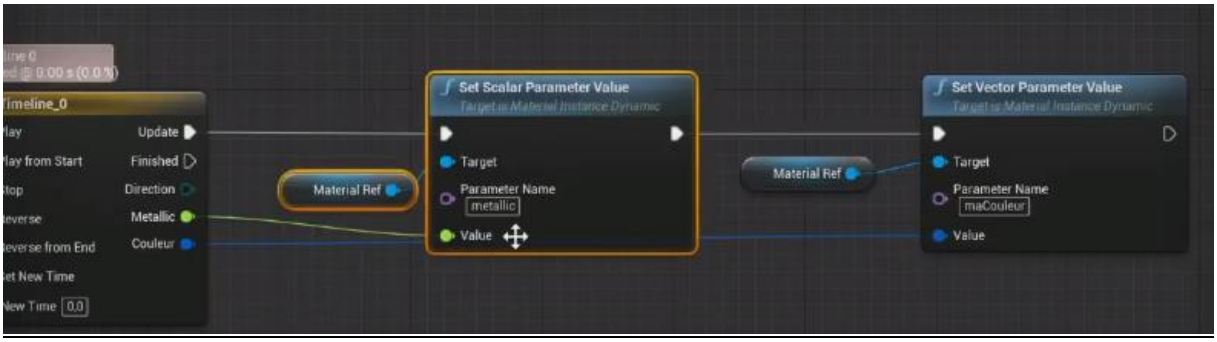
<u>Créer une animation de fondu entre 2 états</u> :

Cliquez sur f+, mettre une key à 0 et renseigner la valeur 1,et mettre une key à 2 et renseigner la valeur 0

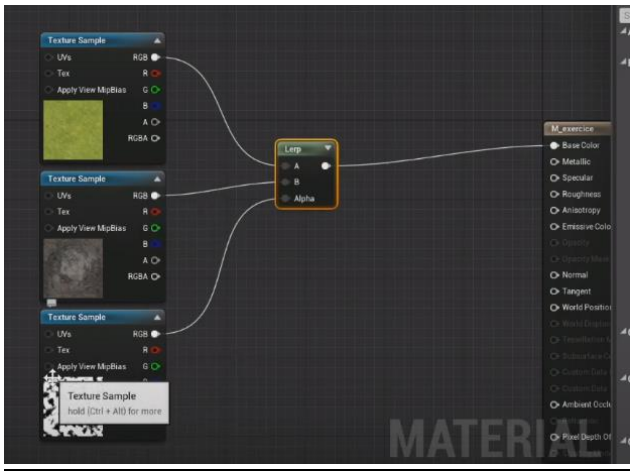


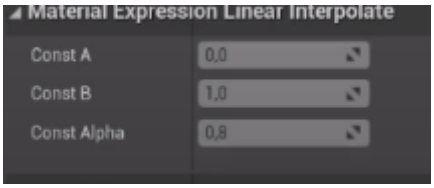
<u>Créer un fondu de couleur en plus :</u>



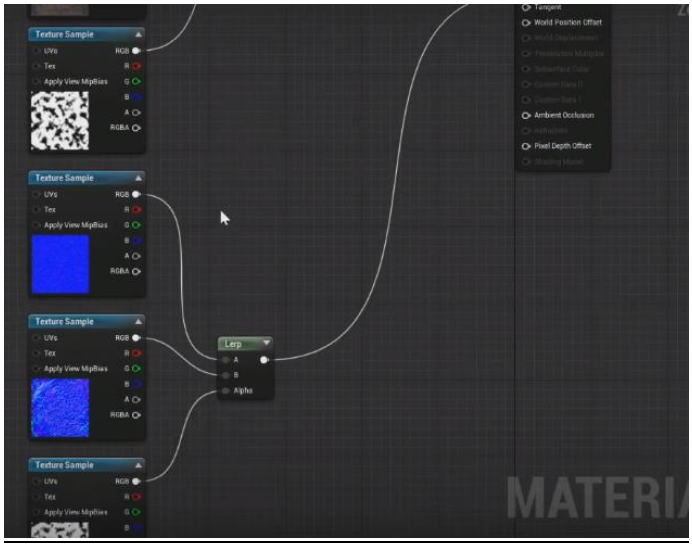


Mixer plusieurs textures :

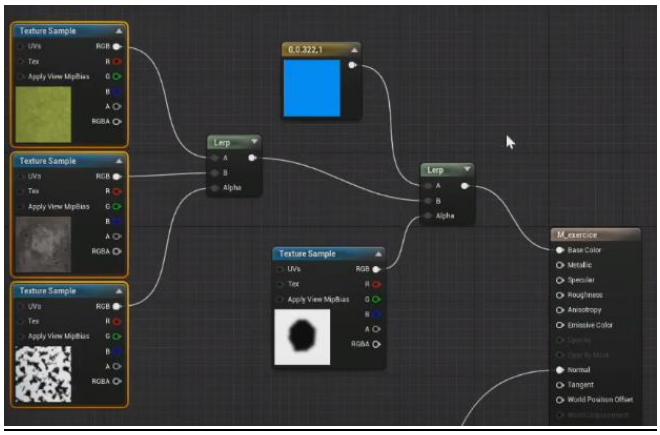




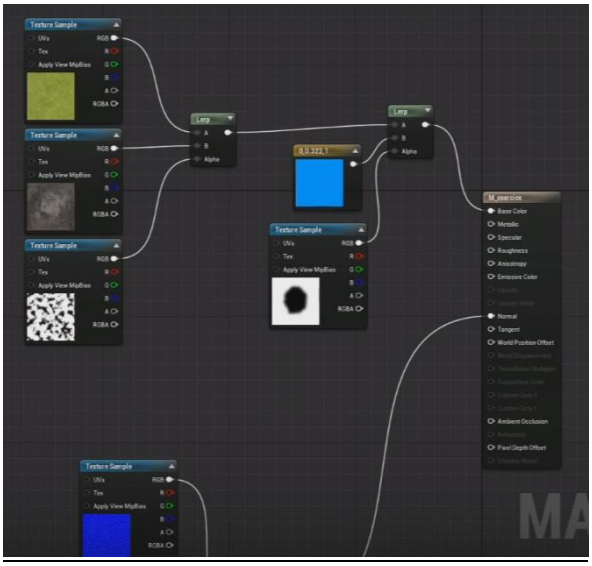
Y ajouter l'effet de profondeur



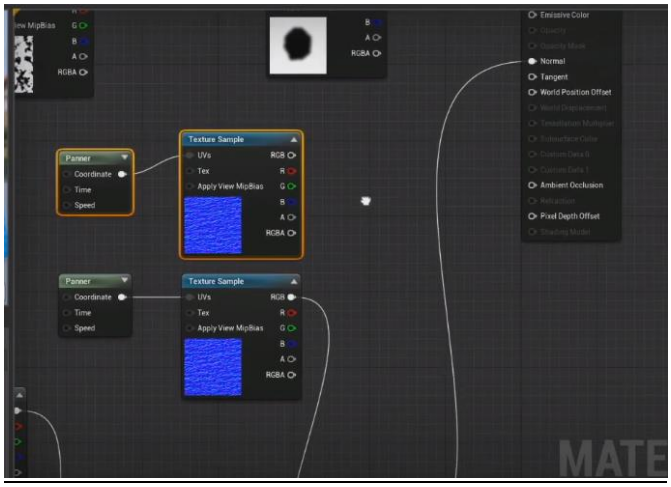
Y ajouter une flaque d’eau

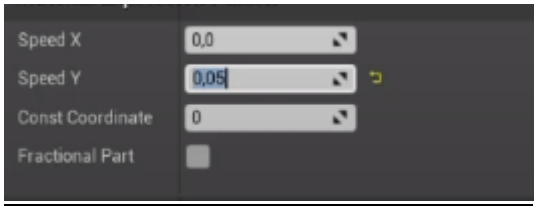


Faire un ilot

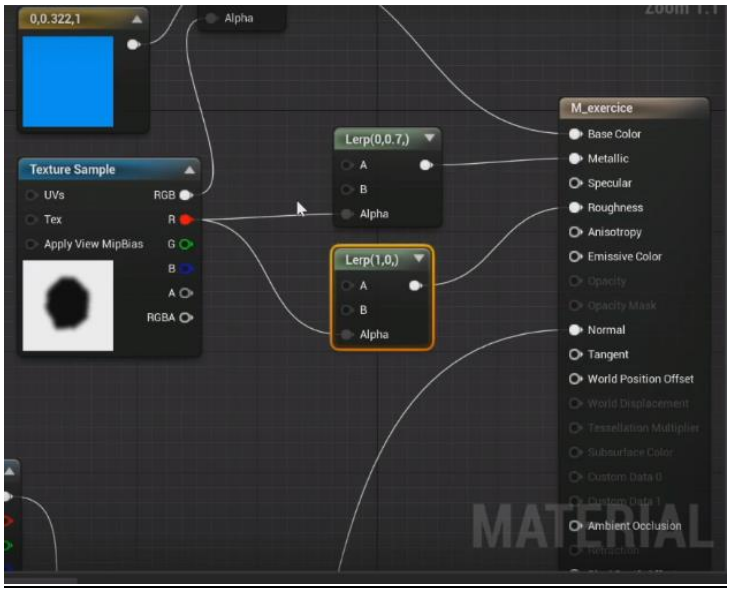


Simuler les vagues

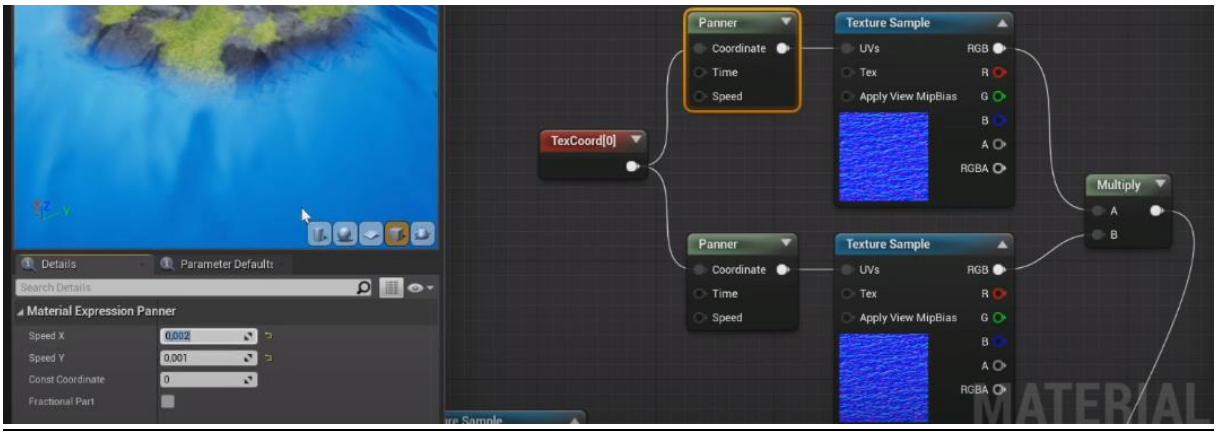




Différencier le metallic et le roughness de l'eau/l'ilot

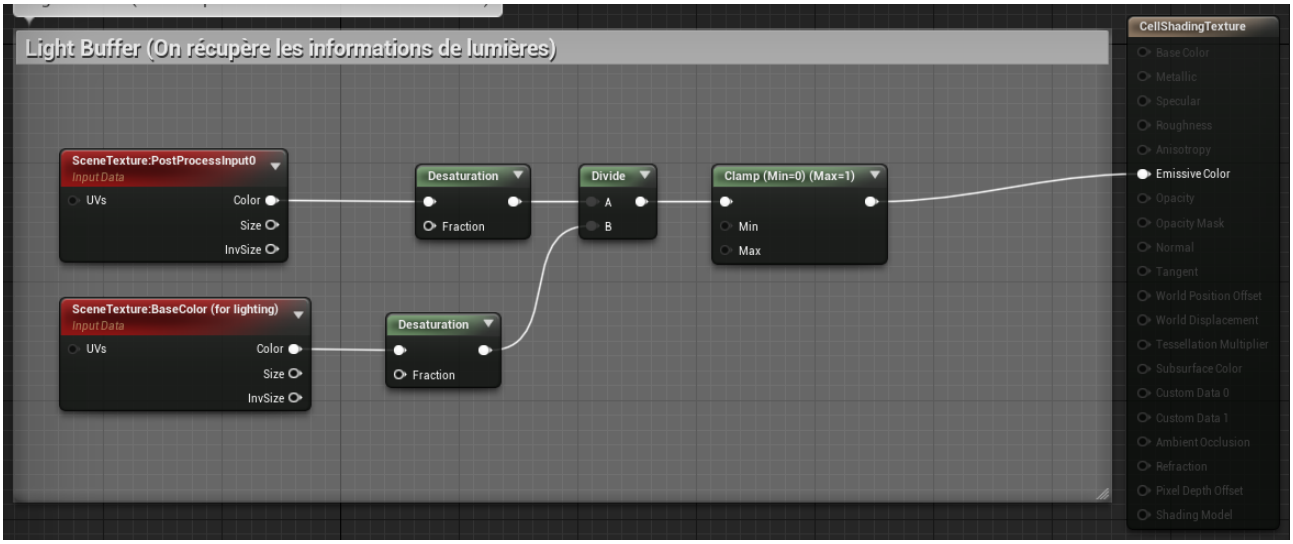


Jouer sur la vitesse des vagues



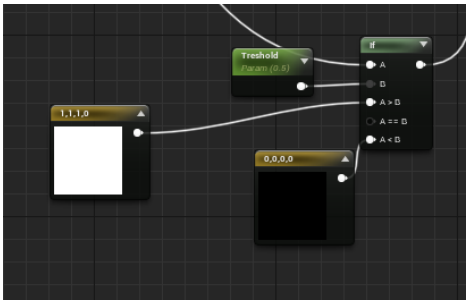
Post-Process Material : Traiter les pixels de l'écran pour obtenir un rendu spécifique
Créer une texture>Material Domain>Post Process>SceneTexture>PostProcessInput0>Color à Emissive Color
Ajout d'un post processing Volume>details>rendering features>Post Process Materials>Array +> Mettre votre material instance>Cocher Infinite Extent(Unbound)

Reproduire ceci dans votre material (pas le Mi !)

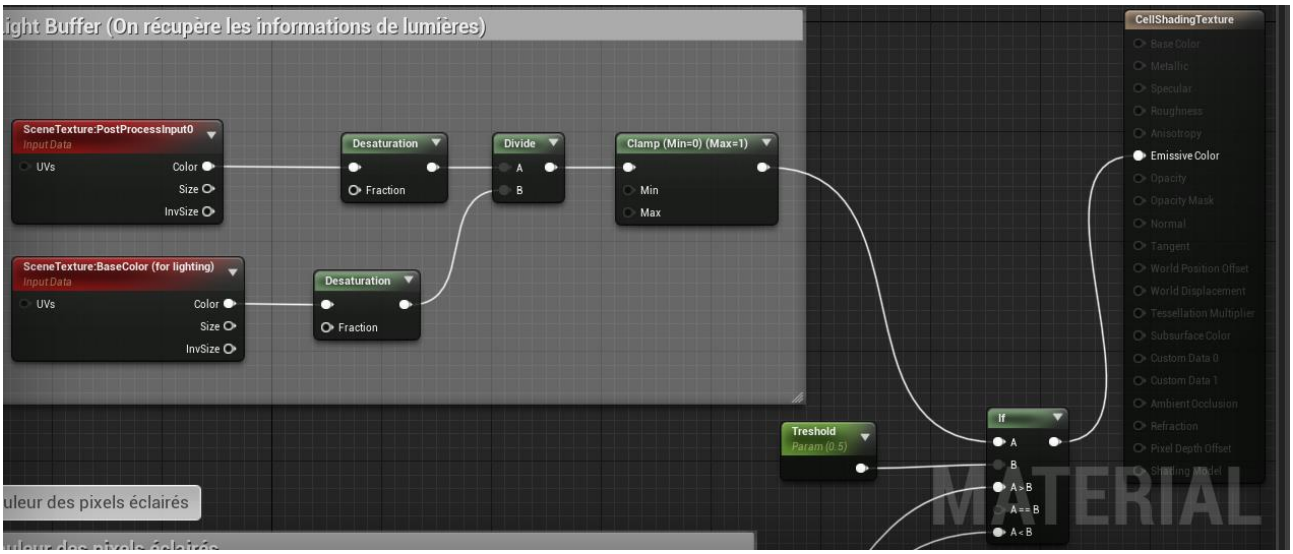


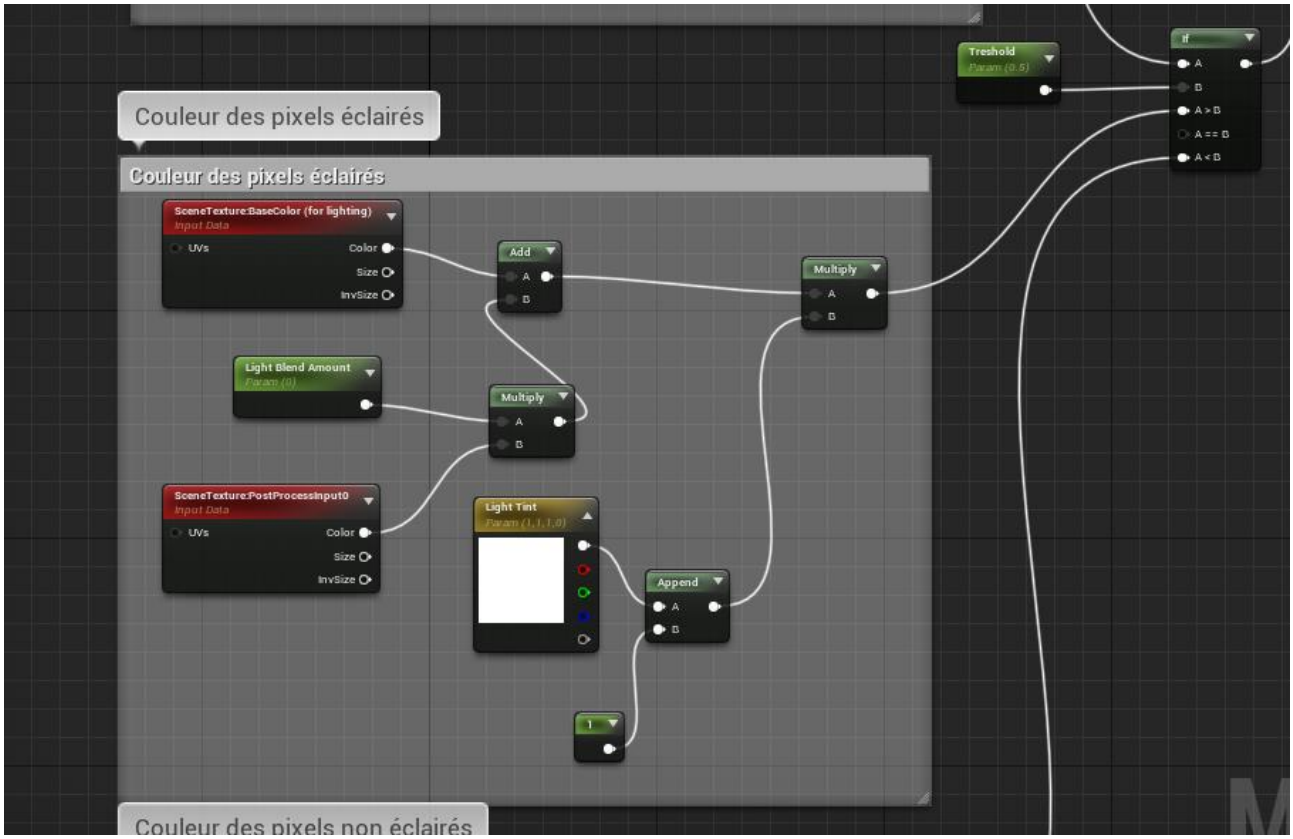
Séléctionnez-le node tout à droite et vous pourrez passez le jeu en noir et blanc !

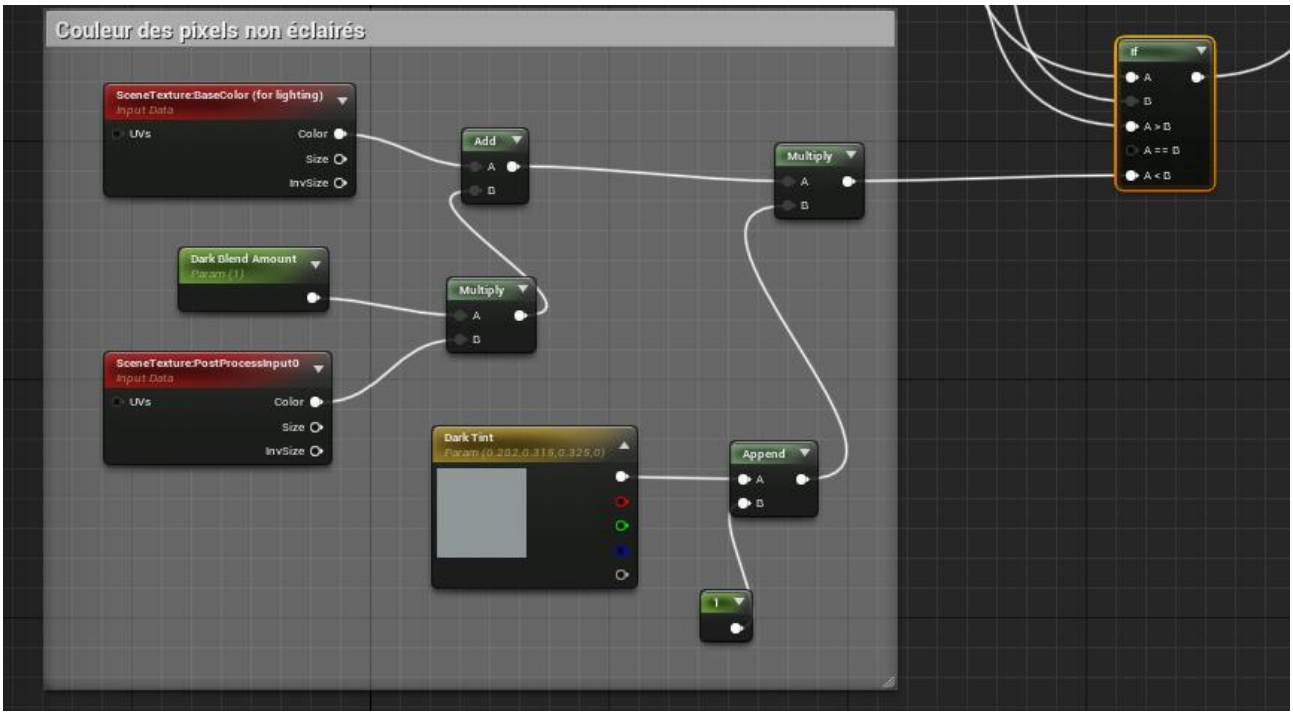
Mettre un scalar nommé Treshold, et 2,4 vectores



Pour les couleurs :







Pour aller plus loin :

A ecrire