



UNREAL ENGINE

Formation sur Unreal Engine 5

développé par Anthony Blanchette-Potvin

pour le Club de développement de jeux vidéo de l'Université Laval



Prérequis.....	2
Disclaimers.....	2
Présentation du produit final.....	3
Création du projet.....	3
Présentation rapide d'Unreal Engine 5	3
Introduction au Blueprint.....	3
Ménage rapide du projet.....	3
Création de l'ennemi au corps-à-corps	4
Création du joueur.....	14
Implémentation des dégâts	20
Création de l'interface utilisateur.....	27
Ajout d'effets visuels et sonores (si le temps le permet)	31
Ajout d'un menu principal (si le temps le permet).....	31
Pour finir.....	32
Annexes.....	33

PRÉREQUIS

- Avoir téléchargé Unreal Engine 5 (voir Annexe 1 – Installation d’Unreal Engine 5)

DISCLAIMERS

- Ce document sert principalement de notes pour la formation sur Unreal Engine 5 présenté par le [Club de développement de jeux vidéo de l’Université Laval](#). Cela dit, la [documentation officielle d’Epic Games](#) est une source d’information beaucoup plus fiable et complète du contenu présenté dans ce document.
- Je ne suis pas un expert d’Unreal Engine 5, alors si vous voulez intervenir, n’hésitez pas.
- Unreal Engine 5 est en *early access*, donc il pourrait être instable.
- Je ne vais pas expliquer en profondeur chacun des outils d’Unreal Engine 5. Le but de la formation est de vous montrer comment faire un jeu vidéo simple dans Unreal Engine 5 et de vous référer vers des ressources qui pourront vous expliquer plus en détails les outils qui vous intéressent.

PRÉSENTATION DU PRODUIT FINAL

CRÉATION DU PROJET

Créer un nouveau projet avec la configuration suivante :

- sélectionner *Top Down*
- sélectionner *Blueprint*
- sélectionner *Desktop* comme *Target Platform*
- sélectionner *Maximum* comme *Quality Preset*
- activer le *Starter Content*
- désactiver *Raytracing*
- avec **NotDiablo** comme *Project Name*

PRÉSENTATION RAPIDE D'UNREAL ENGINE 5

1. panneau *Content Browser*
2. panneau *Viewport*
3. panneau *Menu*
4. panneau *World Outliner*
5. panneau *Details*
6. Blueprint Editor
 - a. panneau *Components*
 - b. panneau *My Blueprint*
 - c. panneau *Menu*
 - d. panneau *Details*
 - e. panneau *Viewport*
 - f. panneau *Event Graph*
 - g. panneau *Construction Script*

INTRODUCTION AU BLUEPRINT

QU'EST-CE QU'UN BLUEPRINT ?

« [...] asset that allows content creators to easily add functionality on top of existing gameplay classes »

ANALYSE D'UN EXEMPLE D'UN BLUEPRINT

MÉNAGE RAPIDE DU PROJET

1. Ménage de **TopDownCharacter**
2. Ménage de **TopDownController**
3. Ménage du *Content Browser*

CRÉATION DE L'ENNEMI AU CORPS-À-CORPS

CRÉATION DU VISUEL

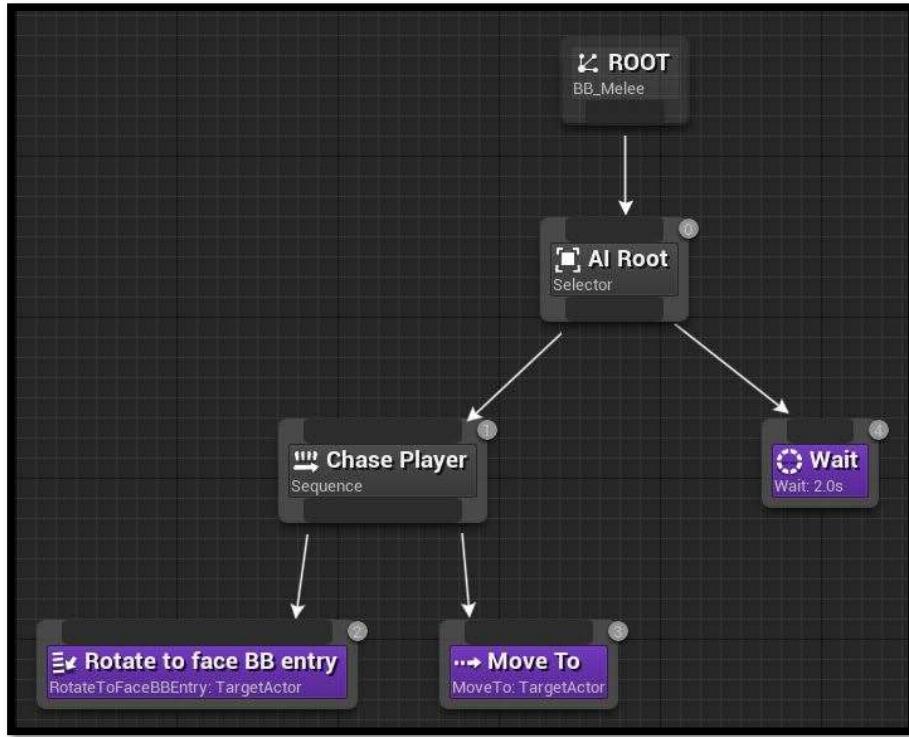
1. Ajouter l'*asset pack* *City of Brass : Enemies* au projet (voir Annexe 2 – Ajout de l'*asset pack* *City of Brass : Enemies* au projet)
2. Dans le dossier *Blueprints*, ajouter un nouveau dossier nommé *Characters*
3. Dans le dossier *Characters*, créer un nouvel *asset* de type *Blueprint Class* :
 - a. avec **Character** comme *classe parente*
 - b. avec **BP_Melee** comme *nom*
4. Ouvrir **BP_Melee**
5. Sélectionner le composant nommé **Mesh** et dans le panneau *Details*, modifier les paramètres suivants :
 - a. *Skeletal Mesh* = **Corpse_Sword**
6. Aller dans le *Viewport* et positionner **Mesh** pour qu'il soit approximativement de la même grosseur que **CapsuleComponent** (ou modifier **CapsuleComponent** pour qu'il soit approximativement de la même grosseur que **Mesh**)
7. Sélectionner le composant nommé **Mesh**, appuyer sur *ADD*, ajouter un composant de type **SkeletalMesh** (s'assurer que le nouveau composant est enfant de **Mesh**) et le nommer **SwordMesh**
8. Sélectionner **SwordMesh** et dans le panneau *Details*, modifier les paramètres suivants :
 - a. *Parent Socket* = **SwordSocket**
 - b. *Skeletal Mesh* = **enemy_Sword_Mesh**
9. Aller dans le *Viewport* et vérifier que le personnage ressemble à l'image ci-dessous :



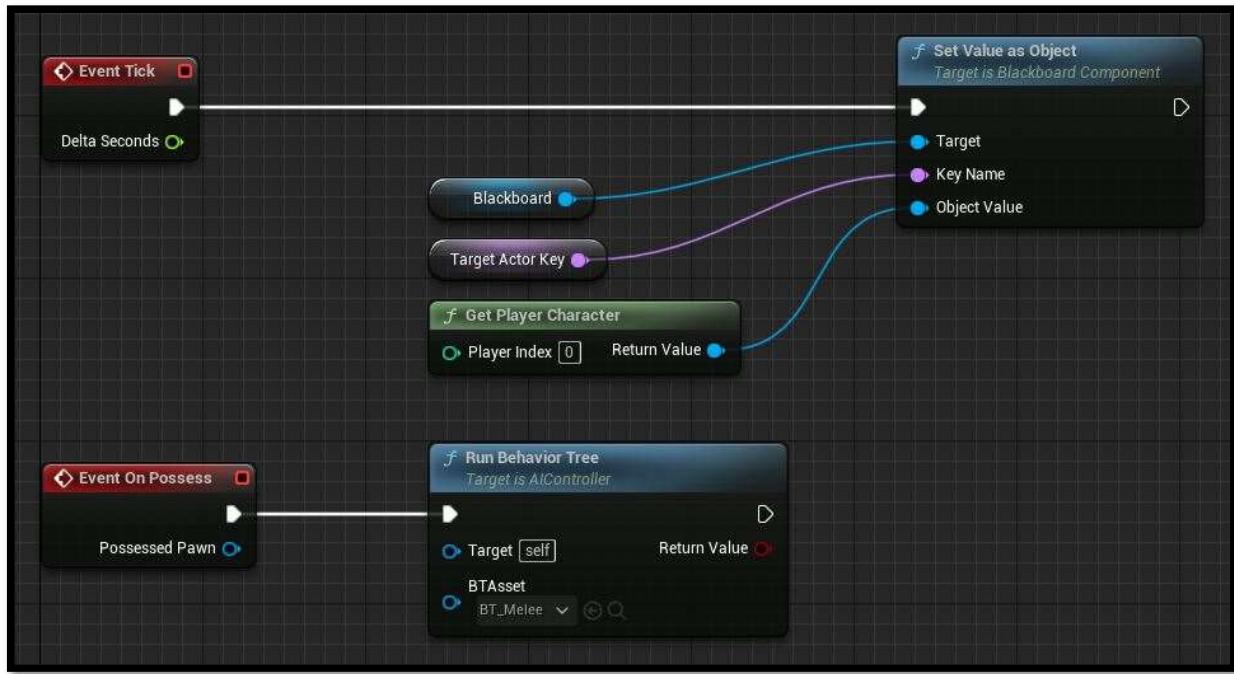
10. Compiler, sauvegarder et fermer **BP_Melee**
11. Ajouter une instance de **BP_Melee** dans le monde en le glissant du *Content Browser* vers le *Viewport* (s'il est glissé sur une surface, il sera automatiquement positionné au-dessus de celle-ci)

AJOUT D'UN AI DE BASE

1. Dans le dossier *Blueprints > Characters*, créer un nouvel asset de type *Blueprint Class* :
 - a. avec **AIController** comme *classe parente*
 - b. avec **BP_MeleeController** comme *nom*
2. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**
3. Dans le panneau *Components*, repérer et sélectionner le composant racine **self**
4. Dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *AI Controller Class* = **BP_MeleeController**
5. Compiler, sauvegarder et fermer **BP_Melee**
6. Dans le dossier *Blueprints > Characters*, créer un nouvel asset de type *Blackboard* :
 - a. avec **BB_Melee** comme *nom*
7. Ouvrir **BB_Melee**
8. Ajouter une clé de type **Object** et la nommer **TargetActor**
9. Sélectionner la clé **TargetActor** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Base Class* = **Actor**
10. Dans le dossier *Blueprints > Characters*, créer un nouvel asset de type *Behavior Tree* :
 - a. avec **BT_Melee** comme *nom*
11. Ouvrir **BT_Melee**
12. Dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Blackboard Asset* = **BB_Melee**
13. Reproduire le graphe ci-dessous :
 - a. pour renommer un nœud, sélectionner le nœud et dans le panneau *Details*, modifier la propriété *Node Name*



14. Sélectionner le nœud **RotateToFaceBBEntry** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Blackboard Key* = **TargetActor**
15. Sélectionner le nœud **MoveTo** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Blackboard Key* = **TargetActor**
16. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_MeleeController**
17. Créer une variable de type **Name** et la nommer **TargetActorKey**
18. Compiler et sauvegarder
19. Sélectionner la variable **TargetActorKey** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Target Actor Key* = **TargetActor**
20. Dans la section *GRAPHS* du panneau *My Blueprint*, repérer et ouvrir le graphe **EventGraph** et reproduire le graphe ci-dessous :

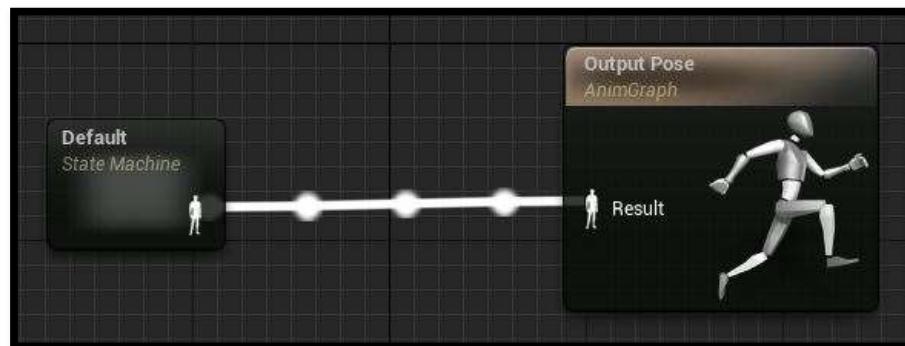


21. Compiler, sauvegarder et fermer **BP_MeleeController**

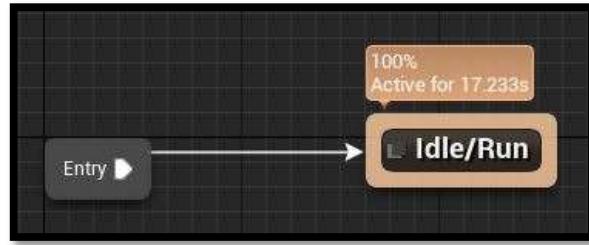
22. Lancer le jeu et vérifier que l'ennemi suit le personnage du joueur

AJOUT DES ANIMATIONS DE BASE

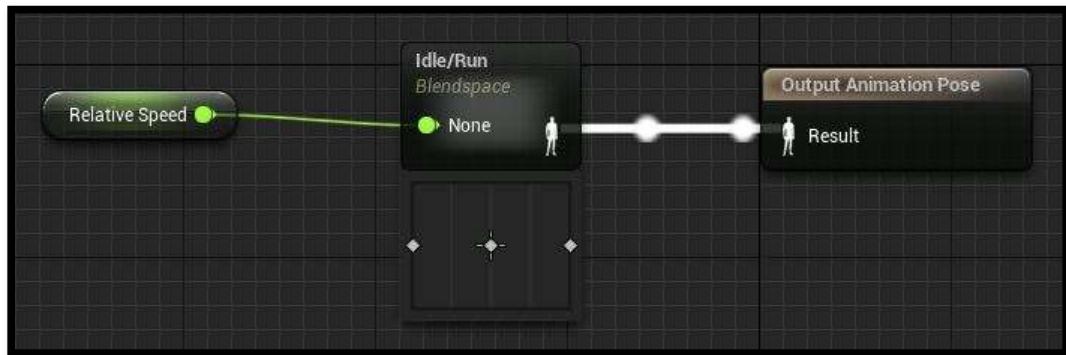
1. Dans le dossier *Animations > Characters*, créer un *asset* de type *Animation Blueprint* :
 - a. avec **AnimInstance** comme *classe parente*
 - b. avec **ABP_Melee** comme *nom*
 - c. avec **Corpse_Skeleton** comme *Target Skeleton*
2. Ouvrir **ABP_Melee**
3. Dans le panneau *My Blueprint*, créer une variable de type **float** et la nommer **RelativeSpeed**
4. Dans le panneau *My Blueprint*, dans la section *ANIMATION GRAPHS*, repérer et ouvrir le graphe **AnimGraph**
5. Reproduire le graphe ci-dessous :



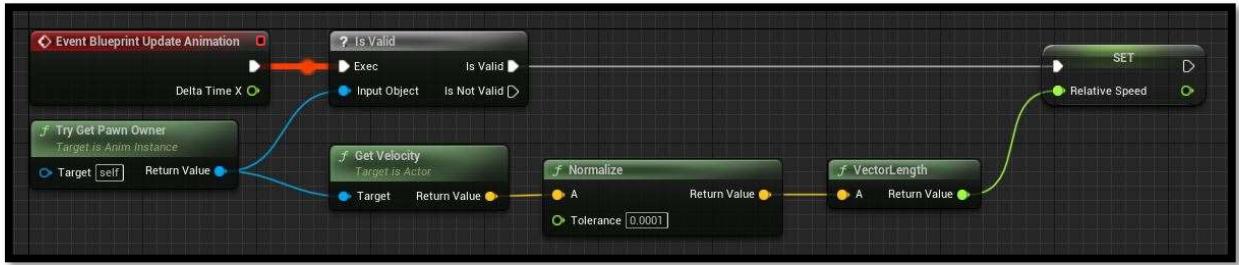
6. Faire un double clic-gauche sur la *state machine Default* pour afficher le graphe de celle-ci
7. Reproduire le graphe ci-dessous :



8. Faire un double clic-gauche sur le *state Idle/Run* pour afficher le graphe de celui-ci
9. Reproduire le graphe ci-dessous :



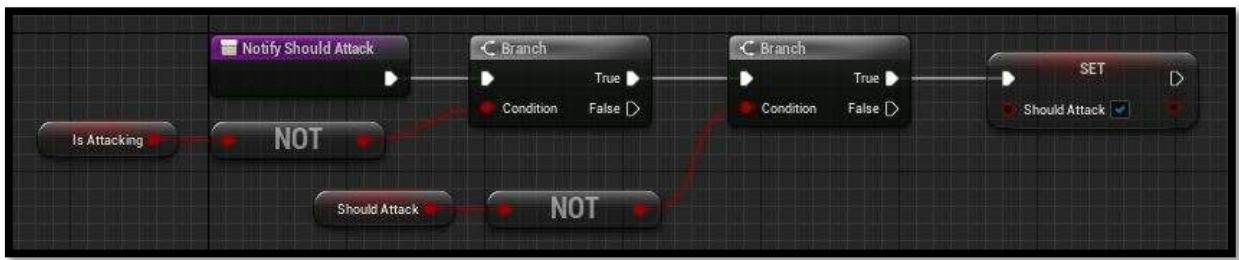
10. Faire un double clic-gauche sur le *blend space Idle/Run* pour afficher le graphe de celui-ci
11. Dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Minimum Axis Value* = **-1**
 - b. *Maximum Axis Value* = **1**
 - c. *Global Interpolation* = **8**
12. Dans le *Content Browser*, rechercher et ajouter les animations suivantes en les glissant dans l'espace du *blend space* :
 - a. **Corpse_alert_idle** en la positionnant au milieu du *blend space*
 - b. **Corpse_Run_Back** en la positionnant à l'extrême gauche du *blend space*
 - c. **Corpse_Run_Forward** en la positionnant à l'extrême droite du *blend space*
13. Compiler et sauvegarder
14. Dans le panneau *My Blueprints*, dans la section *GRAPHS*, repérer et ouvrir **EventGraph**
15. Reproduire le graphe ci-dessous :



16. Compiler, sauvegarder et fermer **ABP_Melee**
17. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**
18. Dans le panneau *Components*, repérer et sélectionner le composant **Mesh**
19. Dans le panneau *Details*, modifier les propriétés suivantes :
20. *Anim Class* = **ABP_Melee**
21. Compiler, sauvegarder et fermer **BP_Melee**
22. Lancer le jeu et vérifier que les animations de l'ennemi sont cohérentes

AJOUT DE LA LOGIQUE D'ATTAQUE

1. Aller dans le dossier *Animations > Characters* et ouvrir **ABP_Melee**
2. Créer deux nouvelles variables :
 - a. **IsAttacking** de type **Boolean**
 - b. **ShouldAttack** de type **Boolean**
3. Compiler et sauvegarder
4. Créer une nouvelle fonction nommée **NotifyShouldAttack** en appuyant sur le + dans la section *FUNCTIONS* du panneau *My Blueprint*
5. Reproduire la structure de nœuds suivante pour la fonction **NotifyShouldAttack** :

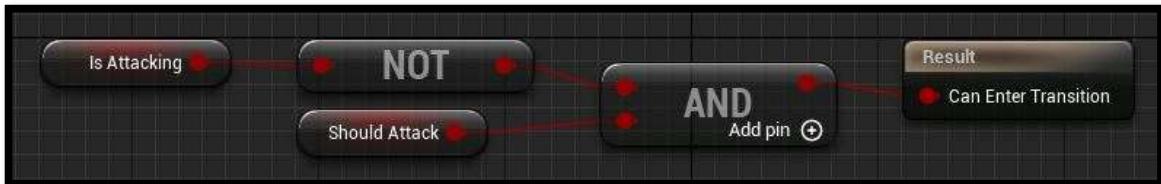


6. Compiler et sauvegarder
7. Dans la section *ANIMATION GRAPHS* du panneau *My Blueprint*, repérer et ouvrir le graphe **Default**
8. Faire un clic-droit dans le *Graph Editor* et ajouter un nouvel état nommé **Attack**
9. Sélectionner l'état **Attack** et dans le panneau *Details*, modifier les paramètres suivants :
 - a. *Entered State Event* = **AttackStarted**
 - b. *Left State Event* = **AttackEnded**
 - c. *Always Reset on Entry* = **true**

10. Ajouter les deux transitions suivantes :

- du noeud **Idle/Run** au noeud **Attack**
- du noeud **Attack** au noeud **Idle/Run**

11. Faire un double-clic sur l'icône de la transition de **Idle/Run** vers **Attack** et reproduire l'image ci-dessous :



12. Faire un double-clic sur l'icône de la transition de **Attack** vers **Idle/Run** et reproduire l'image ci-dessous :



13. Faire un double-clic sur l'état **Attack** et reproduire l'image ci-dessous :



14. Sélectionner le noeud **Play Corpse_Alert_Attack_Basic02** et dans le panneau *Details*, repérer le paramètre *Loop Animation* et le mettre à **false**

15. Compiler, sauvegarder et fermer **ABP_Melee**

16. Dans le *Content Browser*, rechercher l'animation **Corpse_Alert_Attack_Basic02** et l'ouvrir

17. Dans la *timeline*, dérouler le groupe *Notifies* (cela devrait révéler la *track* nommée *1* si ce n'était pas déjà le cas)

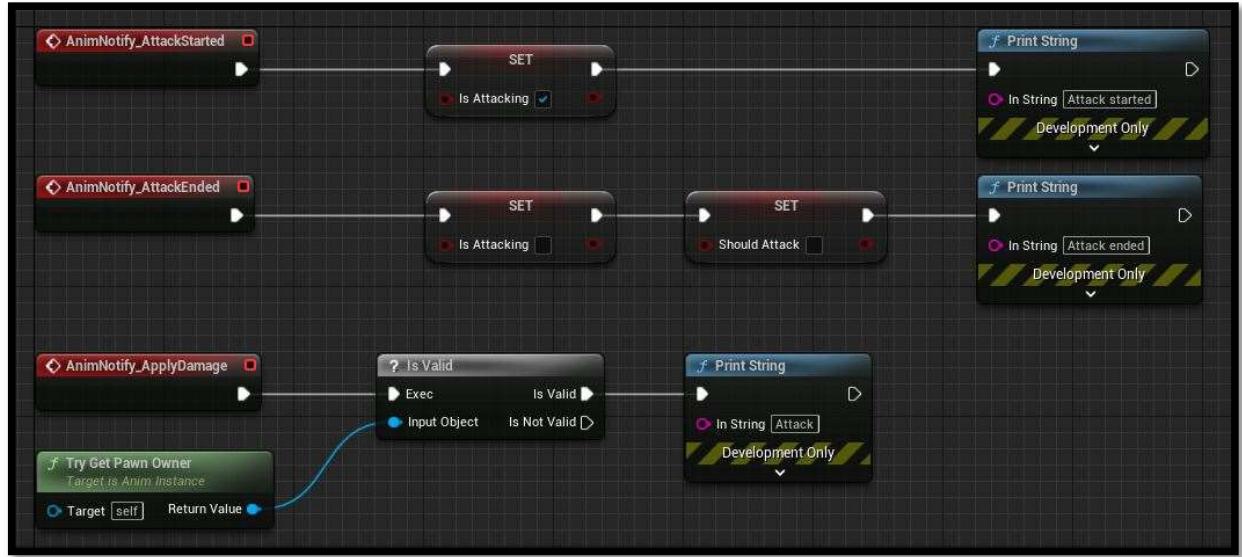
18. Renommer la *track 1* avec le nom *Attack*

19. En s'assurant d'avoir le curseur dans la *track Attack*, faire un clic-droit vis-à-vis le *frame 20*, appuyer sur *Add Notify...*, ensuite *New Notify...* et le nommer **ApplyDamage**

20. Sauvegarder et fermer l'éditeur d'animation

21. Aller dans le dossier *Animations > Characters* et ouvrir **ABP_Melee**

22. Dans la section *GRAPHS* du panneau *My Blueprint*, repérer et ouvrir le graphe **EventGraph** et ajouter les nœuds de l'image ci-dessous :

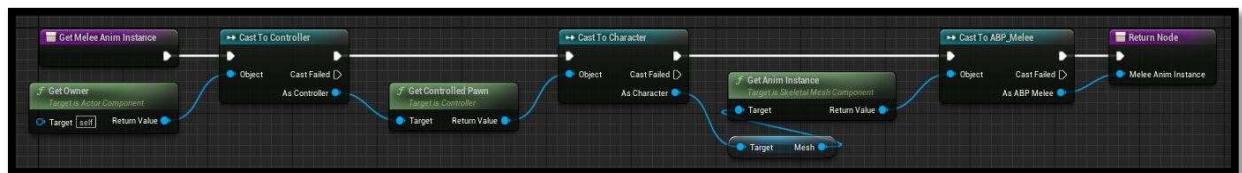


23. Compiler, sauvegarder et fermer **ABP_Melee**

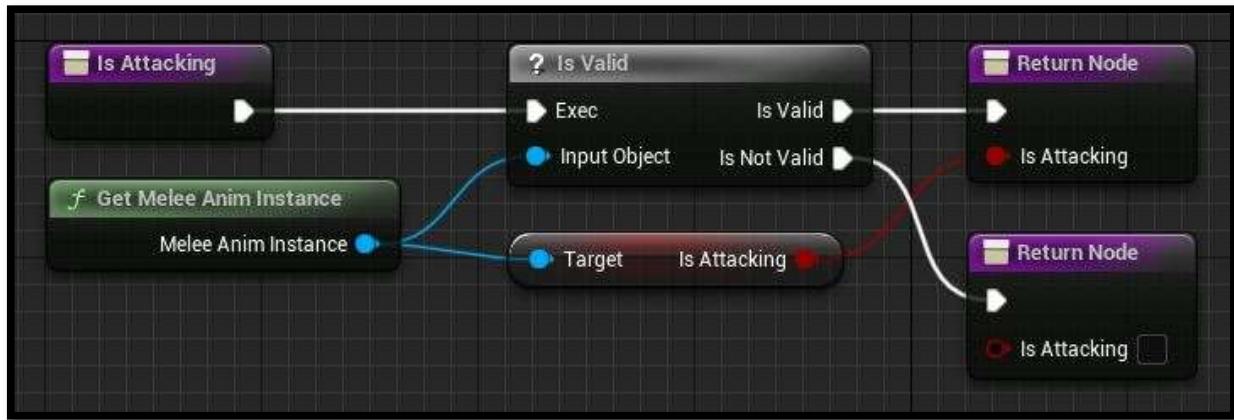
24. Dans le dossier *Blueprints > Components*, créer un nouveau composant de type

ActorComponent nommé **BP_MeleeAnimComponent**

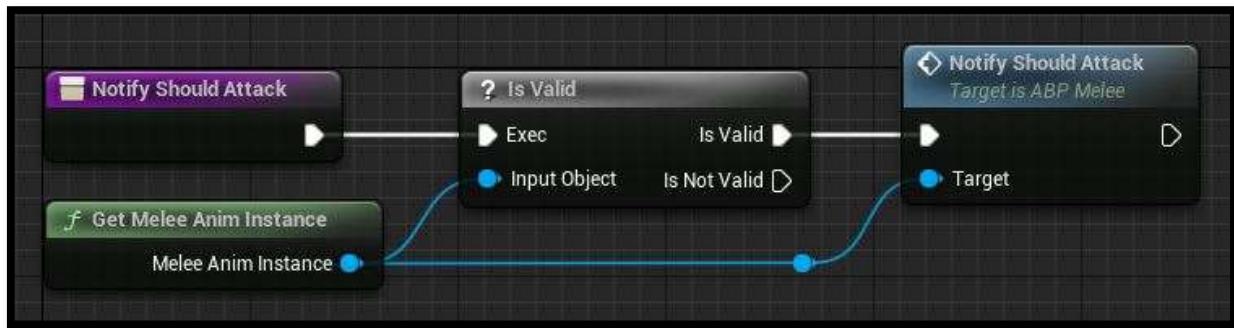
25. Créer une fonction nommée **GetMeleeAnimInstance** et reproduire le graphe ci-dessous dans le corps de la fonction :



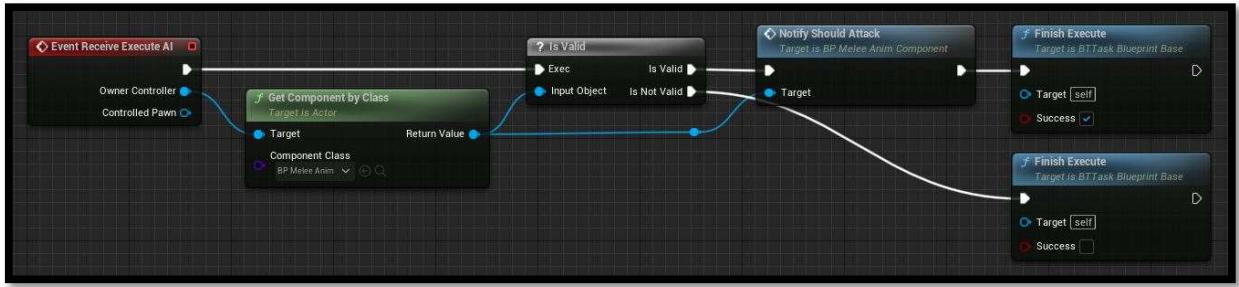
26. Créer une fonction nommée **IsAttacking** et reproduire le graphe ci-dessous dans le corps de la fonction :



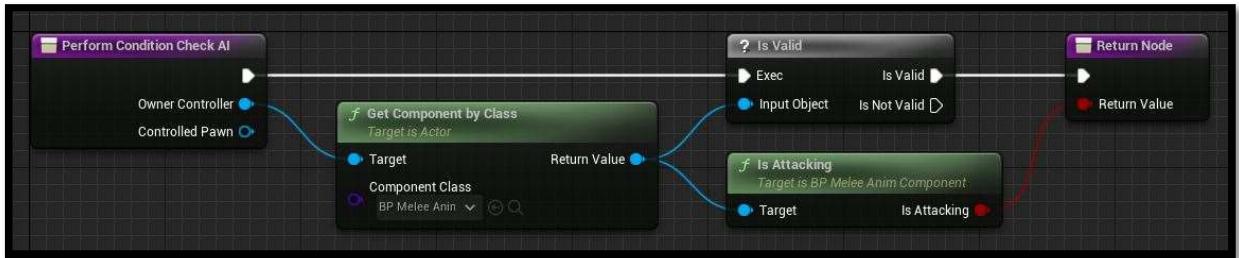
27. Créer une fonction nommée **NotifyShouldAttack** et reproduire le graphe ci-dessous dans le corps de la fonction :



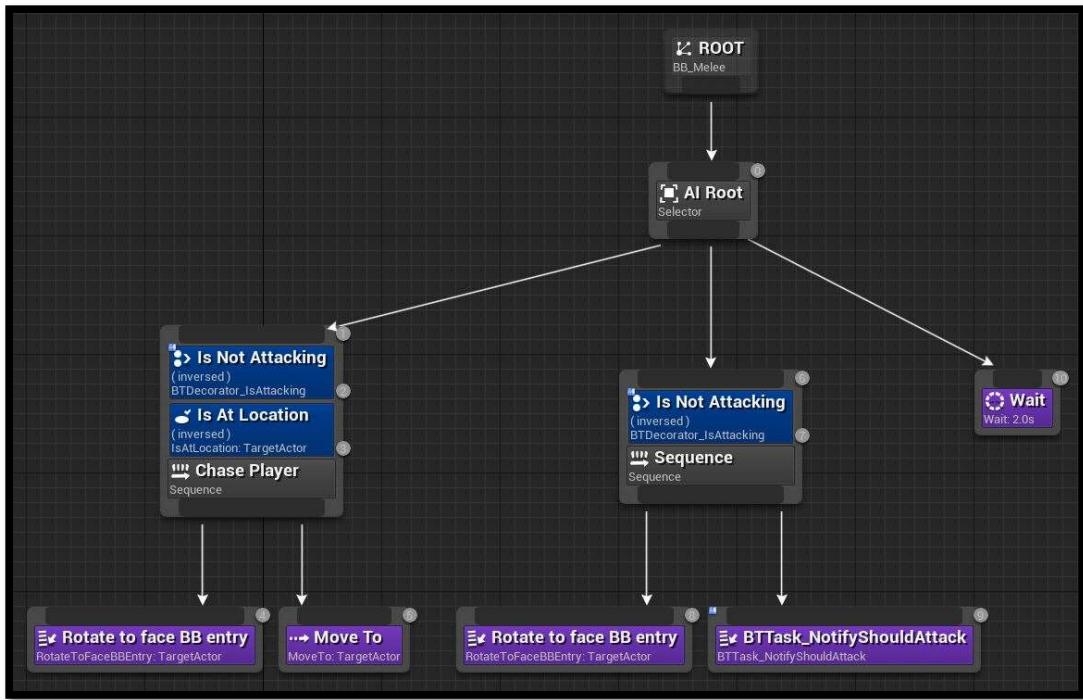
28. Compiler, sauvegarder et fermer **BP_MeleeAnimComponent**
29. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**
30. Ajouter un composant de type **BP_MeleeAnimComponent** et le nommer **MeleeAnimComponent**
31. Compiler, sauvegarder et fermer **BP_Melee**
32. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BT_Melee**
33. Dans le panneau *Menu*, appuyer sur le bouton *New Task* et sélectionner **BTask_BlueprintBase** et la nommer **BTask_NotifyShouldAttack** (pour changer le nom, il faut le faire à partir du *Content Browser*)
34. Reproduire l'image ci-dessous dans **BTask_NotifyShouldAttack** :



35. Compiler, sauvegarder et retourner dans **BT_Melee**
36. Dans le panneau *Menu*, appuyer sur le bouton *New Decorator* et sélectionner **BTDecorator_BlueprintBase** et le nommer **BTDecorator_IsAttacking** (pour changer le nom, il faut le faire à partir du *Content Browser*)
37. Dans **BTDecorator_IsAttacking**, dans le panneau *My Blueprint*, dans la section *FUNCTIONS*, appuyer sur la liste déroulante *Override* et sélectionner **PerformConditionCheckAI**
38. Reproduire l'image ci-dessous dans le corps de la fonction :



39. Compiler, sauvegarder et retourner dans **BT_Melee**
40. Modifier le graphe de **BT_Melee** pour qu'il ressemble à celui dans l'image ci-dessous :
 - a. pour ajouter un décorateur à un nœud, faire un clic-droit sur le nœud, ensuite *Add Decorator...* et choisir le décorateur désiré
 - b. pour inverser la condition d'un décorateur, sélectionner le décorateur à inverser et dans le panneau *Details*, modifier le paramètre *Inverse Condition*



41. Sauvegarder et fermer **BT_Melee**

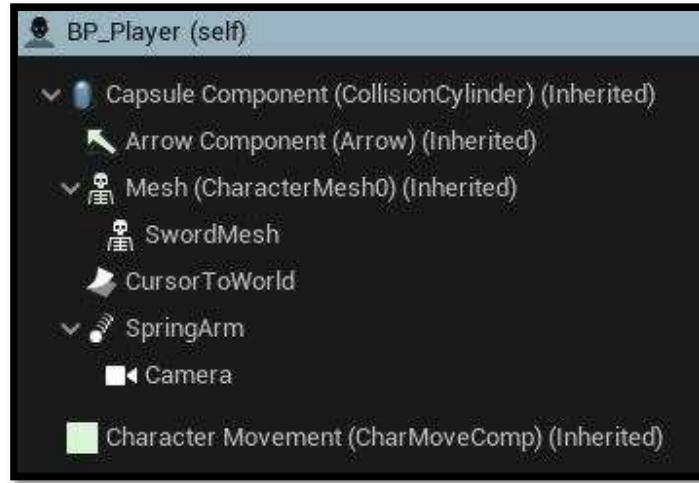
42. Lancer le jeu et vérifier que :

- l'ennemi attaque le joueur lorsqu'il est proche et les trois messages suivants sont affichés dans la console : **Attack started**, **Attack** et **Attack ended**
- l'ennemi reprend la poursuite du joueur même après avoir tenté d'attaquer

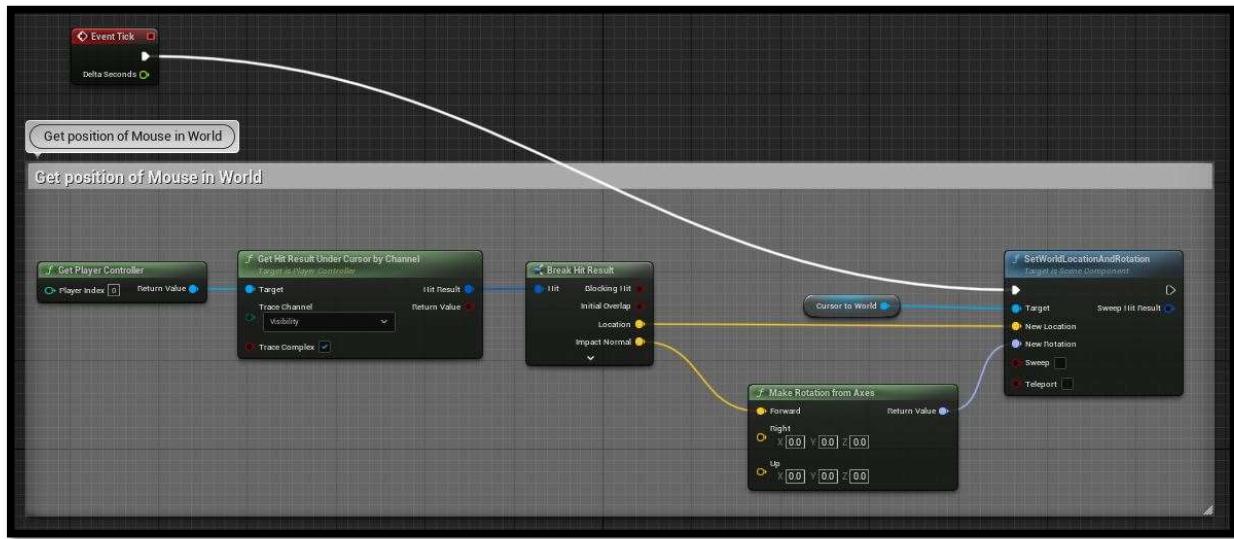
CRÉATION DU JOUEUR

CRÉATION DU BLUEPRINT

- Dans le dossier *Blueprints > Characters*, repérer et dupliquer **BP_Melee** et nommer la copie **BP_Player**
- Dans le dossier *Blueprints > Characters*, repérer et ouvrir **TopDownCharacter**
- Copier les composants suivants : **SpringArm**, **Camera** et **CursorToWorld**
- Repérer et ouvrir **BP_Player**
- Dans le panneau *Components* de **BP_Player**, repérer **CapsuleComponent**, faire un clic-droit sur celui-ci et sélectionner *Paste* pour coller les composants copiés précédemment
- S'assurer que le composant **CursorToWorld** est enfant de **CapsuleComponent**
- Vérifier que la hiérarchie de composants ressemble à l'image ci-dessous :

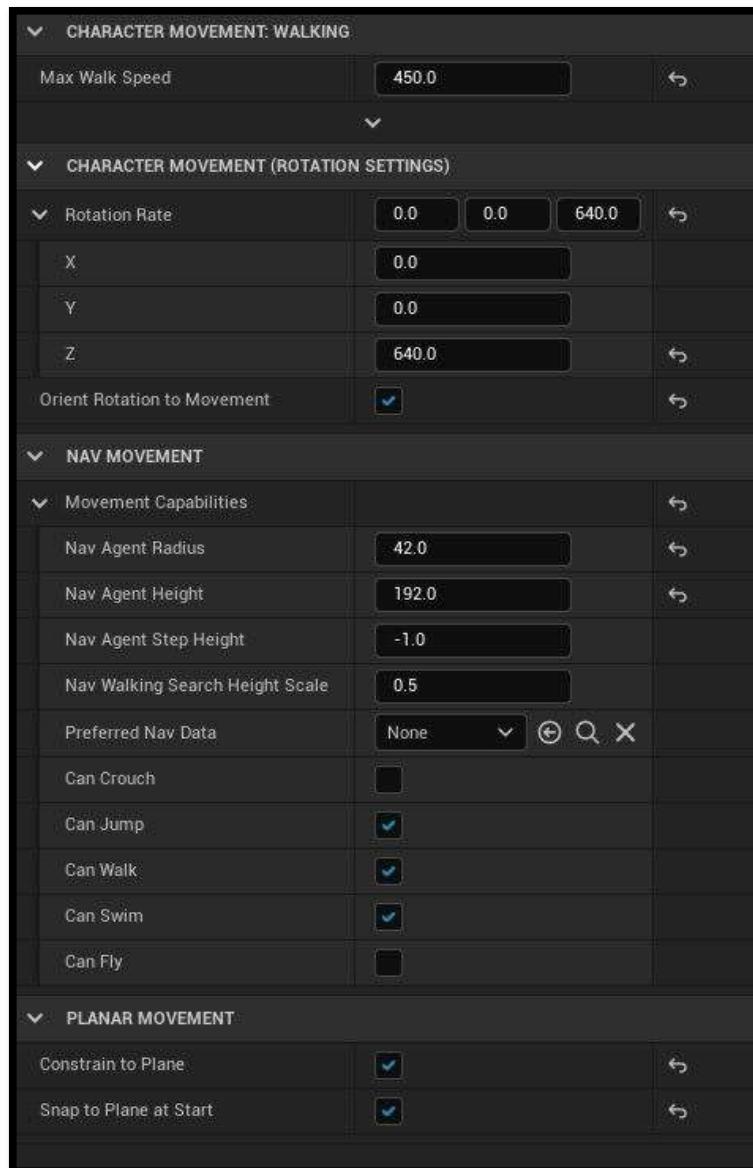


8. Retourner dans **TopDownCharacter**
9. Dans le panneau *My Blueprint*, dans la section *GRAPHS*, repérer et ouvrir le graphe **EventGraph**
10. Sélectionner et copier tous les nœuds du graphe
11. Retourner dans **BP_Player**
12. Dans le panneau *My Blueprint*, dans la section *GRAPHS*, repérer et ouvrir le graphe **EventGraph**
13. Sélectionner et supprimer tous les nœuds du graphe
14. Coller les nœuds collés précédemment
15. Vérifier que le graphe ressemble à l'image ci-dessous :



16. Retourner dans **TopDownCharacter**
17. Dans le panneau *Components*, repérer le composant **CharacterMovement** et le sélectionner

18. Dans le panneau *Details*, appuyer sur et sélectionner l'option *Show Only Modified Properties*
19. Retourner dans **BP_Player**
20. Dans le panneau *Components*, repérer le composant **CharacterMovement** et le sélectionner
21. S'assurer que les valeurs des propriétés sont exactement les mêmes que celui de **TopDownCharacter** à l'exception de :
- Max Walk Speed* = **450**
22. Vérifier que le panneau *Details* de **BP_Player** ressemble à l'image ci-dessous (avec l'option *Show Only Modified Properties* activée) :



23. Dans le panneau *Components*, repérer le composant **Mesh** et modifier les propriétés suivantes :

a. *Skeletal Mesh* = **Corpse_Melee**

24. Aller dans le *Viewport* et vérifier que le personnage ressemble à l'image ci-dessous :



25. Compiler, sauvegarder et fermer **BP_Player** et **TopDownCharacter**

26. Dans le panneau *World Outliner*, repérer et supprimer l'acteur **TopDownCharacter**

27. Dans le dossier *Blueprints*, repérer et ouvrir **TopDownGameMode**

28.Modifier les propriétés suivantes :

a. *Default Pawn Class* = **BP_Player**

29. Compiler, sauvegarder et fermer **TopDownGameMode**

30. Appuyer sur *Play* et vérifier que le personnage *spawn* correctement et est contrôlé par le joueur

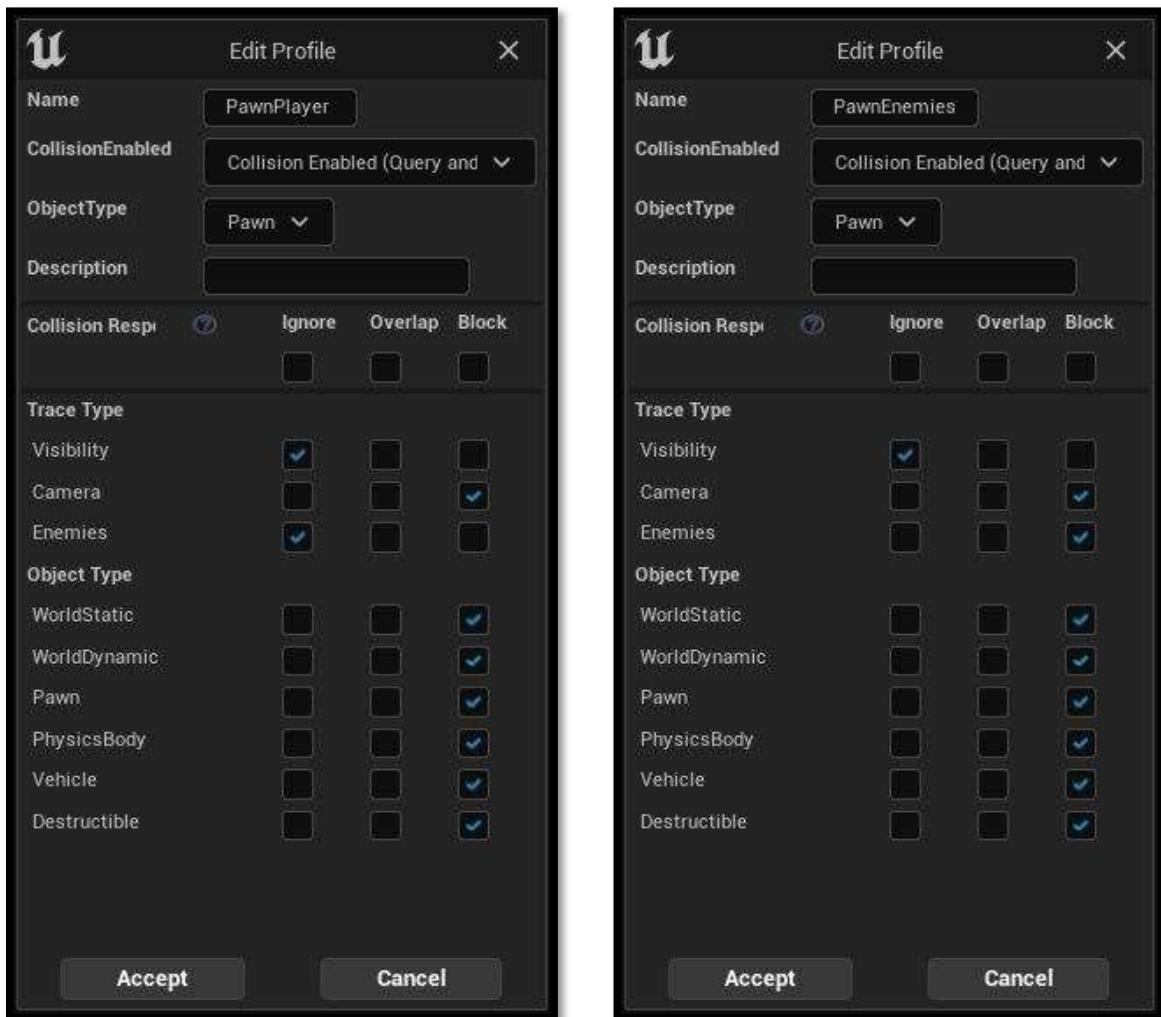
SUPPRESSION DU MANNEQUIN

1. Dans le dossier *Blueprints > Characters*, repérer et supprimer **TopDownCharacter**
2. Dans le dossier *Animations*, repérer et supprimer le dossier *Mannequin* et tout son contenu
3. Dans le dossier *Materials*, repérer et supprimer le dossier *Mannequin* et tout son contenu
4. Dans le dossier *Meshes*, repérer et supprimer le dossier *Mannequin* et tout son contenu
5. Dans le dossier *Textures*, repérer et supprimer le dossier *Mannequin* et tout son contenu

6. Sauvegarder
7. Lancer le jeu et vérifier que tout fonctionne encore

CRÉATION DE CANAUX DE COLLISIONS

1. Ouvrir les paramètres du projet (*Edit > Project Settings...*)
2. Dans la barre de recherche, écrire : « trace channels »
3. Appuyer sur le bouton *New Trace Channel...*
4. Créer un nouveau *trace channel* avec les propriétés suivantes :
 - a. *Name* = **Enemies**
 - b. *Default Response* = *Ignore*
5. Dans la barre de recherche, écrire : « presets »
6. Appuyer sur le bouton *New...*
7. Créer deux nouveaux *presets* identiques à ceux dans l'image ci-dessous :

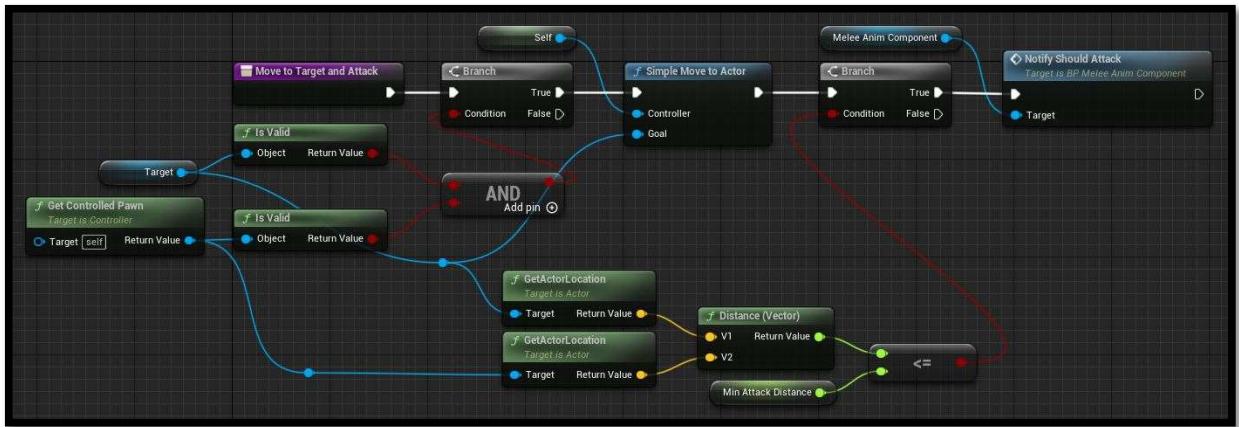


8. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Player**

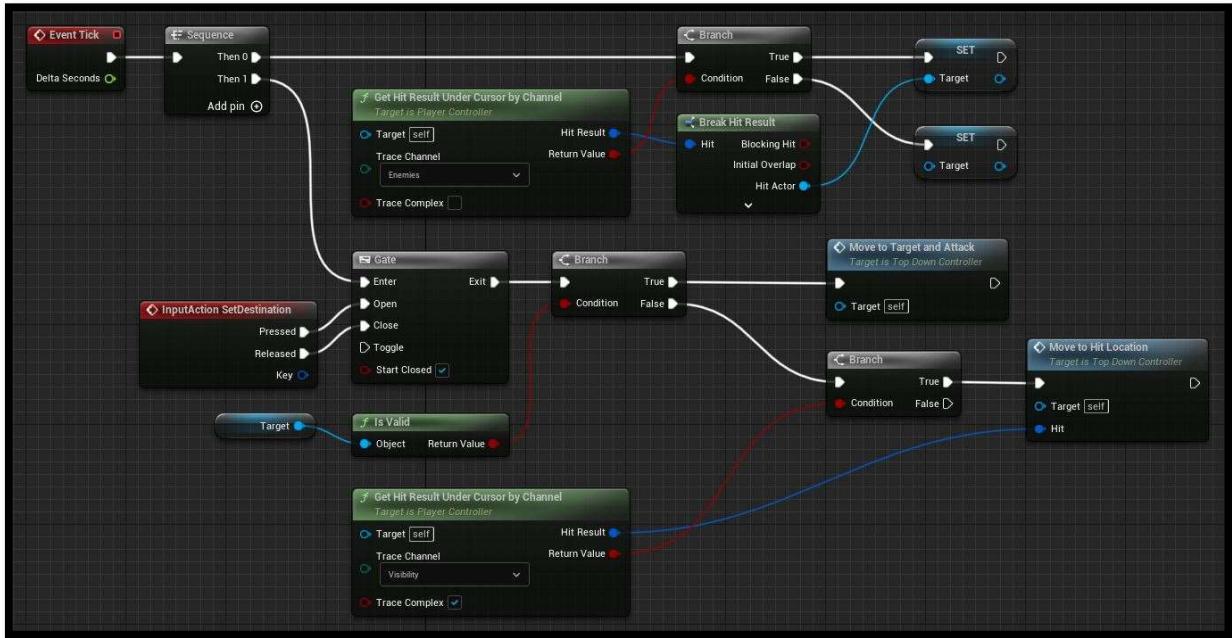
9. Sélectionner le composant **CapsuleComponent** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Collision Presets* = **PawnPlayer**
10. Compiler, sauvegarder et fermer **BP_Player**
11. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**
12. Sélectionner le composant **CapsuleComponent** et dans le panneau *Details*, modifier les propriétés suivantes :
 - a. *Collision Presets* = **PawnEnemies**
13. Compiler, sauvegarder et fermer **BP_Melee**

AJOUT DE LA LOGIQUE D'ATTAQUE

1. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **TopDownController**
2. Créer une nouvelle variable nommée **MinAttackDistance** de type **float**
3. Sélectionner la nouvelle variable et dans le panneau *Details*, modifier les propriétés suivantes :d
 - a. *Min Attack Distance* = **150**
4. Créer une nouvelle variable nommée **Target** de type **Actor**
5. Ajouter un composant de type **BP_MeleeAnimComponent** et le nommer **MeleeAnimComponent**
6. Créer une nouvelle fonction appelée **MoveToTargetAndAttack**
7. Dans le corps de la nouvelle fonction, reproduire la structure de nœuds du graphe ci-dessous :



8. Compiler et sauvegarder
9. Dans le panneau *My Blueprint*, dans la section *GRAPHS*, repérer et ouvrir le graphe **EventGraph**
10. Supprimer tous les nœuds
11. Reproduire le graphe ci-dessous :



12. Compiler, sauvegarder et fermer [TopDownController](#)

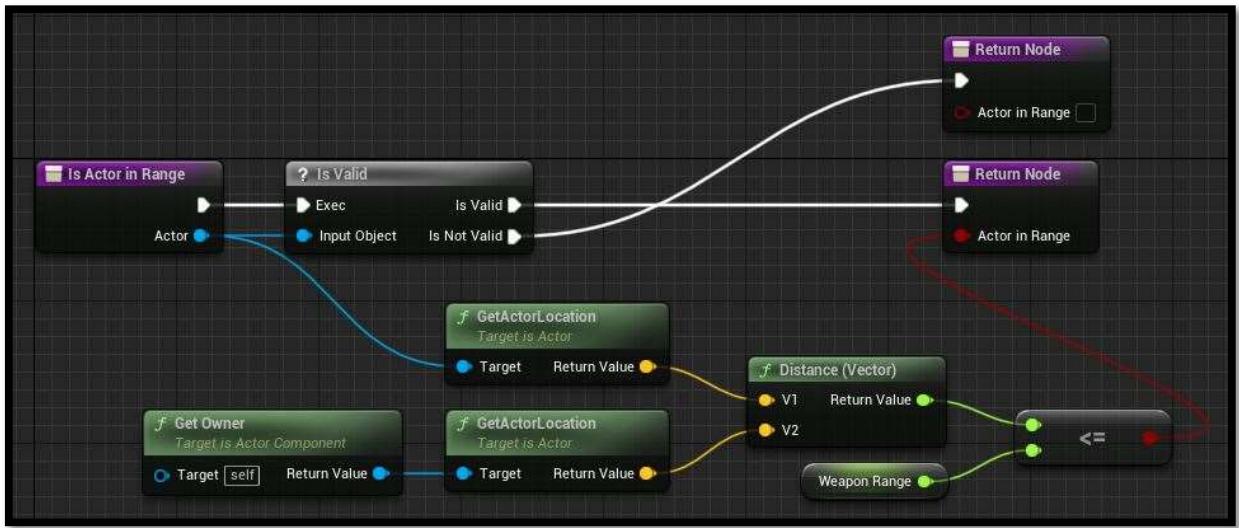
13. Lancer le jeu et vérifier que lorsqu'un clic-gauche est fait sur l'ennemi :

- le personnage se déplace vers l'ennemi s'il est trop loin
- le personnage attaque l'ennemi s'il est assez proche et les trois messages suivants sont affichés dans la console : [Attack started](#), [Attack](#) et [Attack ended](#)

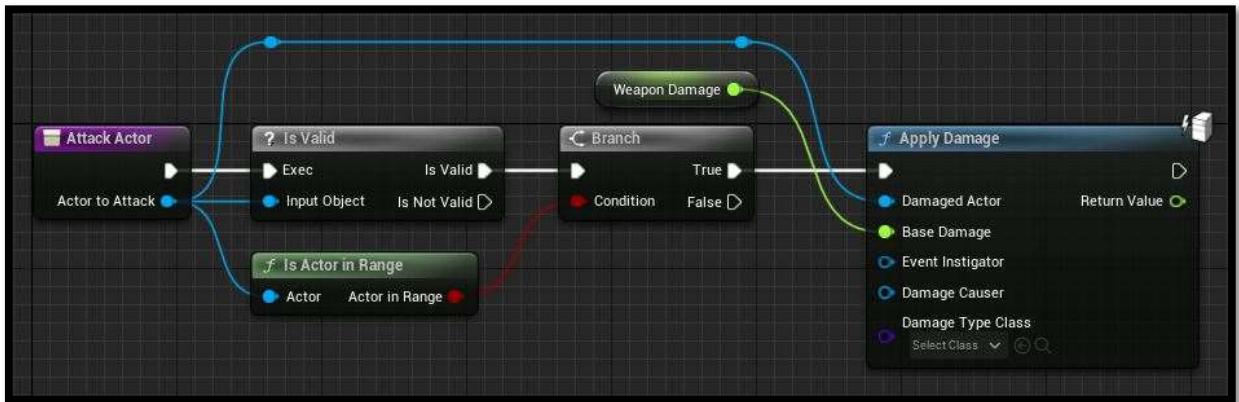
IMPLEMENTATION DES DÉGATS

CRÉATION DU COMPOSANT D'ARME

- Dans le dossier *Blueprints > Components*, créer un nouveau composant de type [ActorComponent](#) nommé [BP_WeaponComponent](#)
- Ouvrir [BP_WeaponComponent](#)
- Ajouter deux nouvelles variables :
 - [weaponDamage](#) de type [float](#) avec valeur par défaut de [15](#)
 - [weaponRange](#) de type [float](#) avec valeur par défaut de [150](#)
- Compiler et sauvegarder
- Ajouter une nouvelle fonction pure nommée [IsActorInRange](#) et reproduire le graphe ci-dessous dans le corps de la fonction :



- Ajouter une nouvelle fonction nommée **AttackActor** et reproduire le graphe ci-dessous dans le corps de la fonction :

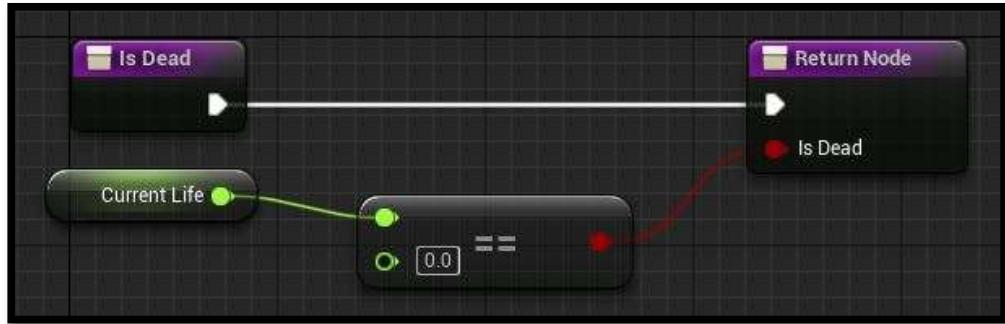


- Compiler, sauvegarder et fermer **BP_WeaponComponent**
- Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**
- Ajouter un composant de type **BP_WeaponComponent** et le nommer **WeaponComponent**
- Compiler, sauvegarder et fermer **BP_Melee**
- Faire de même avec **BP_Player**

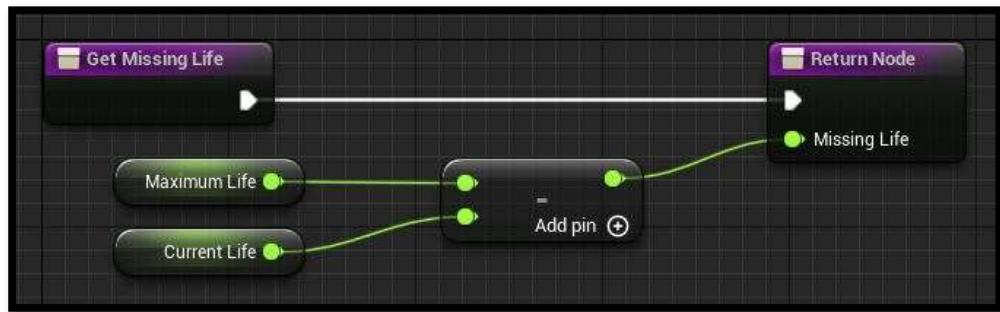
CRÉATION DU COMPOSANT DE VIE

- Dans le dossier *Blueprints > Components*, créer un nouveau composant de type **ActorComponent** nommé **BP_HealthComponent**
- Ouvrir **BP_HealthComponent**
- Ajouter deux nouvelles variables :
 - maximumLife** de type **float** avec valeur par défaut de **100**

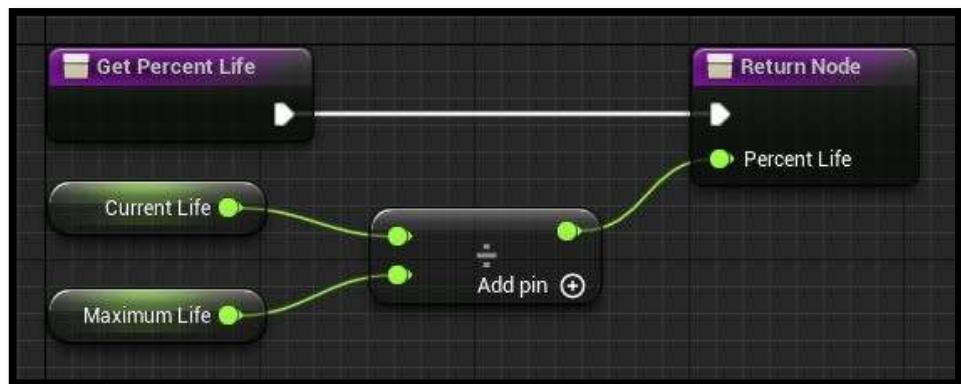
- b. `currentLife` de type `float` avec valeur par défaut de `100`
4. Créer un nouvel *event dispatcher* et le nommer `OnDie`
 5. Créer un nouvel *event dispatcher* et le nommer `OnLifeChanged`
 6. Compiler et sauvegarder
 7. Créer une nouvelle fonction pure nommée `IsDead` et reproduire le graphe ci-dessous dans le corps de la fonction :



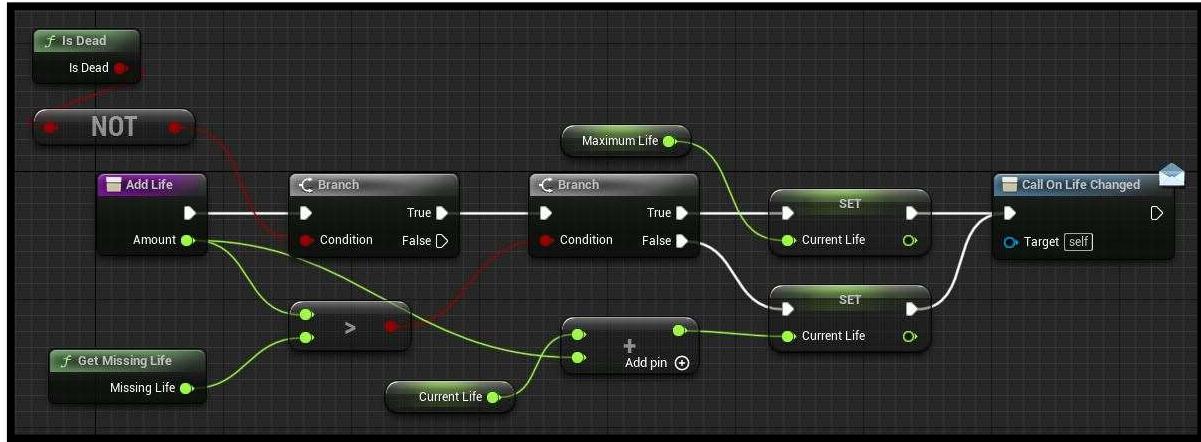
8. Créer une nouvelle fonction pure nommée `GetMissingLife` et reproduire le graphe ci-dessous dans le corps de la fonction :



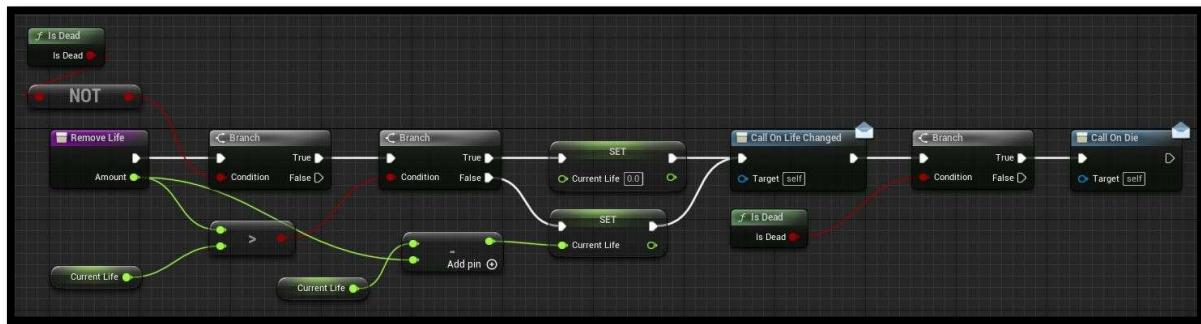
9. Créer une nouvelle fonction pure nommée `GetPercentLife` et reproduire le graphe ci-dessous dans le corps de la fonction :



10. Créer une nouvelle fonction nommée **AddLife** et reproduire le graphe ci-dessous dans le corps de la fonction :



11. Créer une nouvelle fonction nommée **RemoveLife** et reproduire le graphe ci-dessous dans le corps de la fonction :



12. Compiler, sauvegarder et fermer **BP_HealthComponent**

13. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_Melee**

14. Ajouter un composant de type **BP_HealthComponent** et le nommer **HealthComponent**

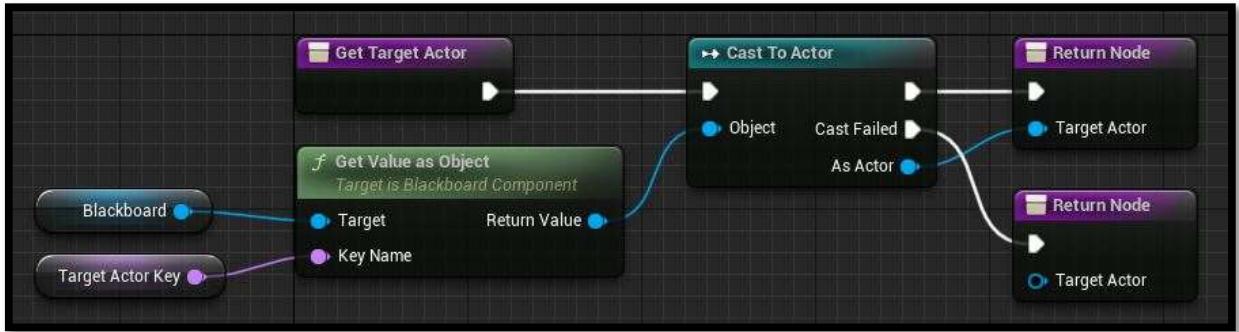
15. Compiler, sauvegarder et fermer **BP_Melee**

16. Faire de même avec **BP_Player**

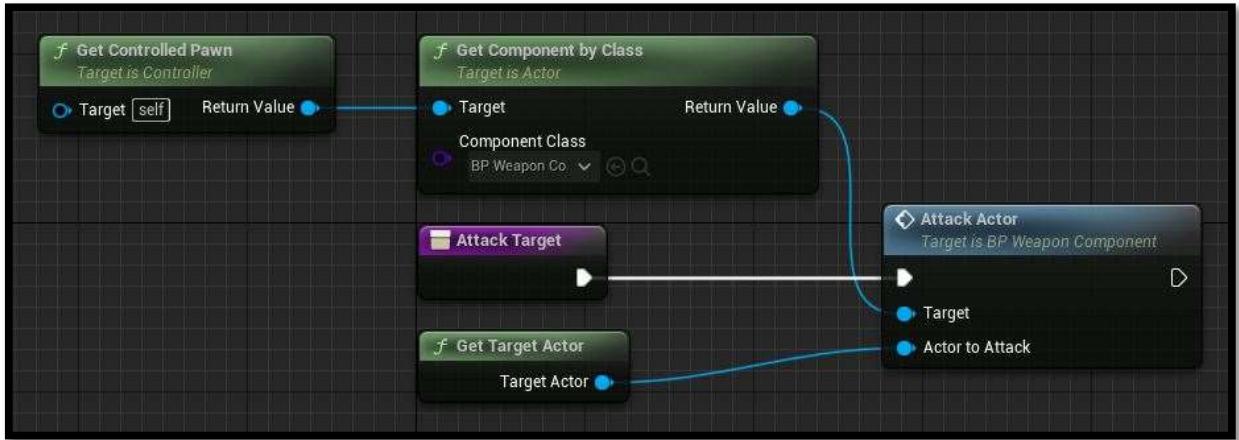
CONFIGURATION DES CONTRÔLEURS

1. Dans le dossier *Blueprints > Characters*, repérer et ouvrir **BP_MeleeController**

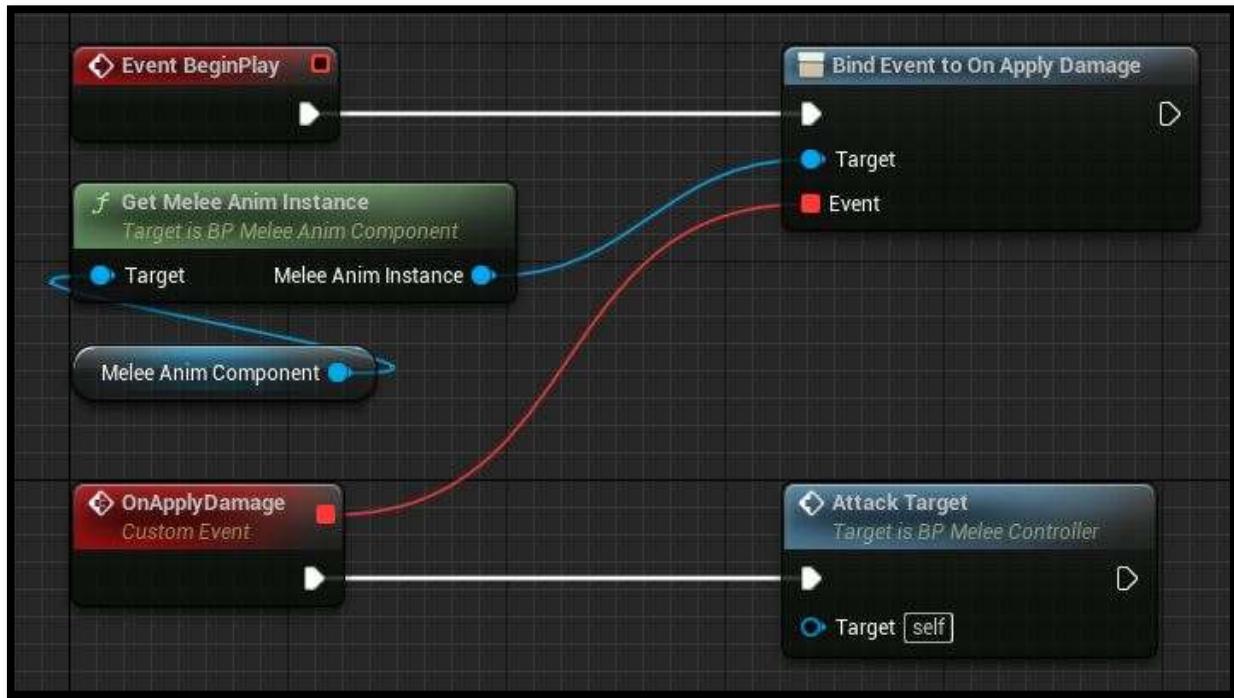
2. Ajouter une nouvelle fonction pure nommée **GetTargetActor** et reproduire le graphe ci-dessous :



3. Ajouter une nouvelle fonction nommée **AttackTarget** et reproduire le graphe ci-dessous :



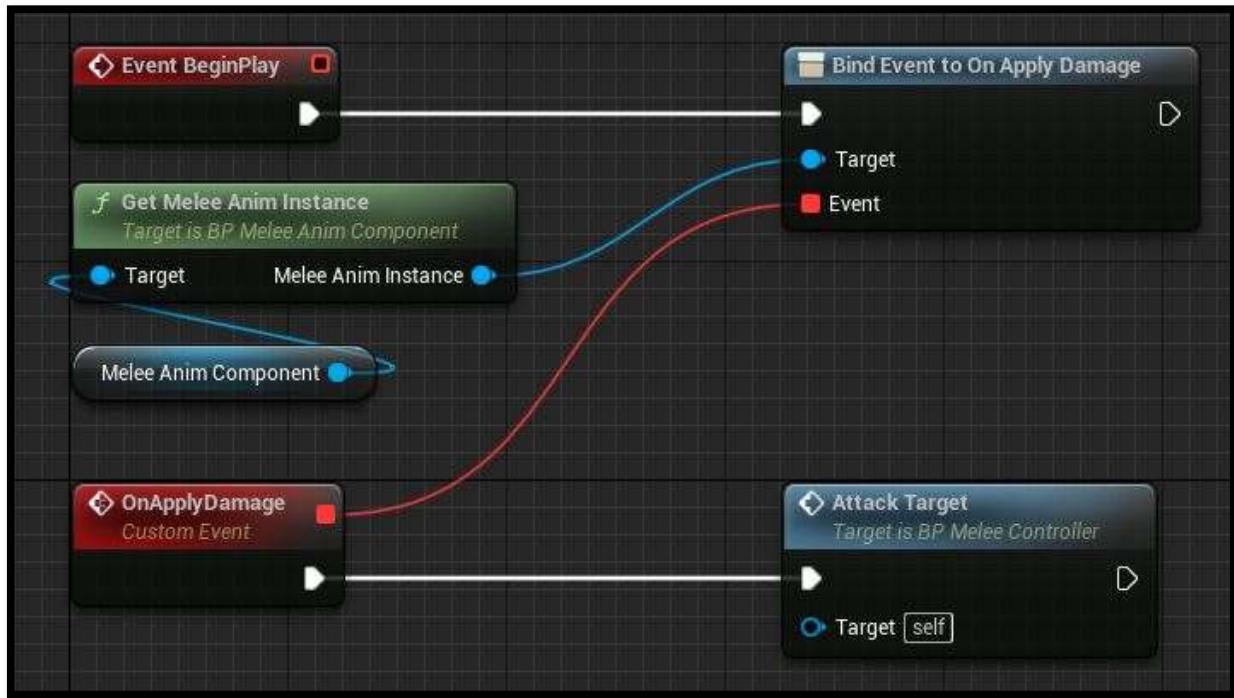
4. Dans le panneau *My Component*, dans la section *GRAPHS*, repérer et ouvrir **EventGraph**
5. Ajouter les nœuds suivants dans le graphe :
- faire un clic-droit dans le graphe et *Add Custom Event...* pour ajouter un évènement *custom*



6. Compiler, sauvegarder et fermer [BP_MeleeController](#)
7. Dans le dossier *Blueprints > Characters*, repérer et ouvrir [TopDownController](#)
8. Ajouter une nouvelle fonction nommée [AttackTarget](#) et reproduire le graphe ci-dessous :



9. Dans le panneau *My Component*, dans la section *GRAPHS*, repérer et ouvrir [EventGraph](#)
10. Ajouter les nœuds suivants dans le graphe :
 - a. faire un clic-droit dans le graphe et *Add Custom Event...* pour ajouter un évènement custom



11. Compiler, sauvegarder et fermer [TopDownController](#)

APPLICATION DES DÉGATS

1. Dans le dossier *Blueprints > Characters*, repérer et ouvrir [BP_Melee](#)
2. Dans le panneau *My Component*, dans la section *GRAPHS*, repérer et ouvrir [EventGraph](#)
3. Ajouter les nœuds suivants dans le graphe :



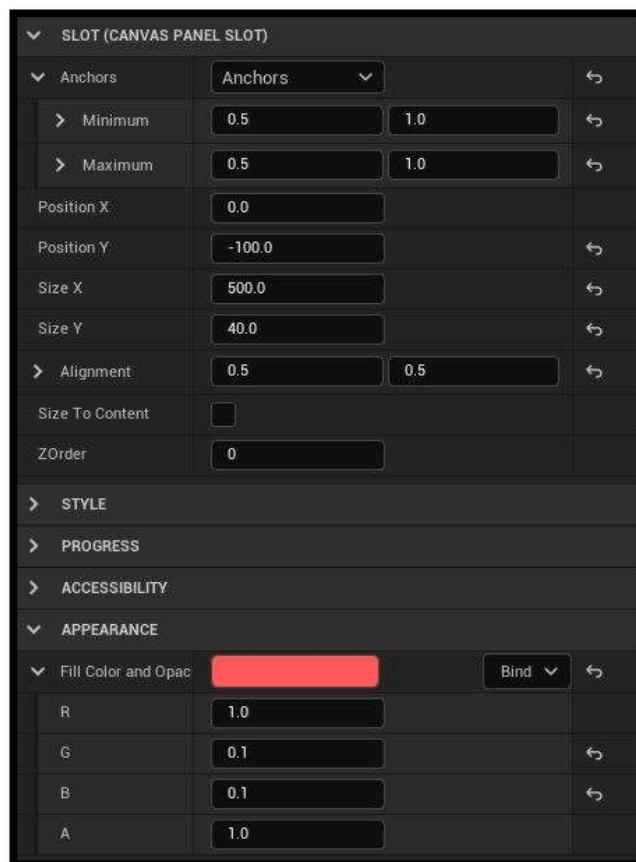
4. Compiler, sauvegarder et fermer [BP_Melee](#)

5. Faire de même avec **BP_Player**
6. Lancer le jeu et vérifier que :
 - a. après quelques attaques, l'ennemi est détruit
 - b. après quelques attaques de l'ennemi, le joueur est détruit

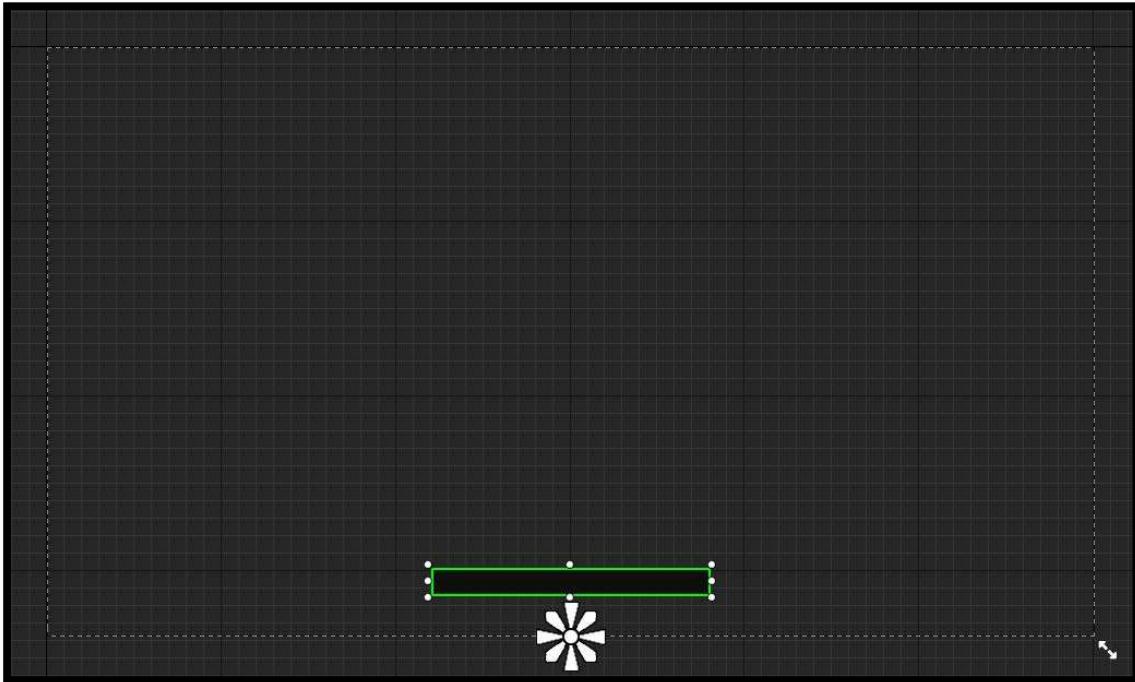
CRÉATION DE L'INTERFACE UTILISATEUR

CRÉATION DU WIDGET

1. Dans le *Content Browser*, créer un nouveau dossier nommé *UI*
2. Dans le dossier *UI*, créer un nouvel asset de type *Widget Blueprint* et le nommer **WBP_PlayerHealth**
3. Ouvrir **WBP_PlayerHealth**
4. À partir du panneau *Palette*, dans la section *COMMON*, repérer la *progress bar* et la glisser dans le *Viewport*
5. À partir du panneau *Hierarchy*, repérer et renommer la *progress bar* avec le nom **HealthBar**
6. Sélectionner **HealthBar**
7. Dans le panneau *Details*, modifier les propriétés pour qu'elles soient identiques à celle dans l'image ci-dessous :



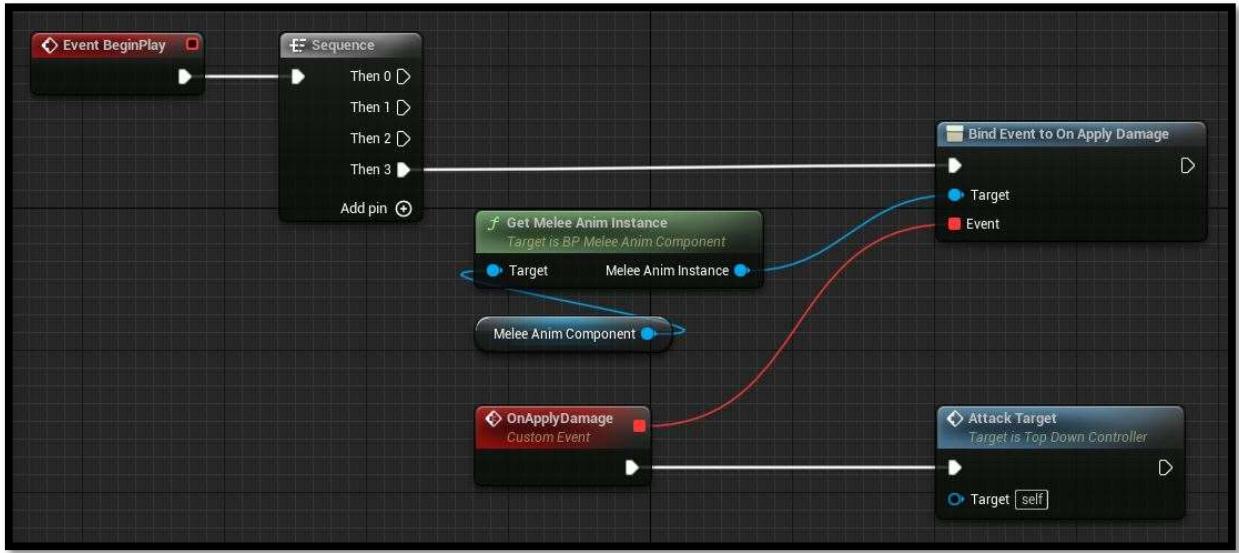
- Vérifier que **HealthBar** à la même position et est de même taille que celle dans l'image ci-dessous :



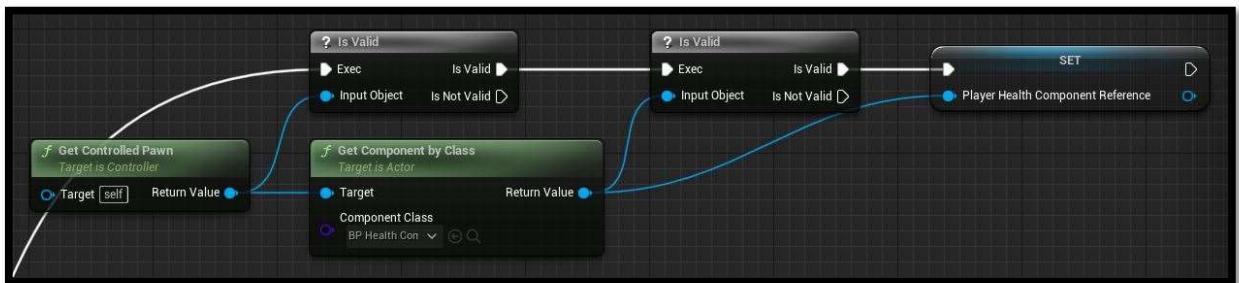
- Compiler, sauvegarder et fermer **WBP_PlayerHealth**

MODIFICATION DU CONTRÔLEUR DU JOUEUR

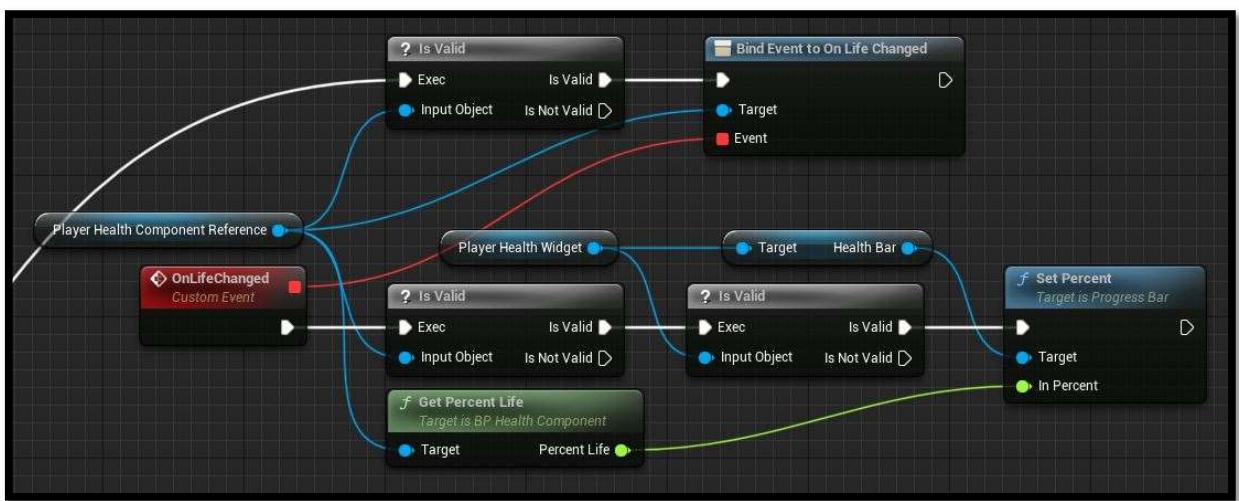
- Dans le dossier *Blueprints > Characters*, repérer et ouvrir **TopDownController**
- Ajouter les deux nouvelles variables suivantes :
 - PlayerHealthWidget** de type **WBP_PlayerHealth**
 - PlayerHealthComponentReference** de type **BP_HealthComponent**
- Dans le panneau *My Blueprint*, dans la section *GRAPHS*, repérer et ouvrir **EventGraph**
- Ajouter un nœud **Sequence** tout juste après l'évènement **BeginPlay** de sorte que le graphe ressemble maintenant à celui de l'image ci-dessous :



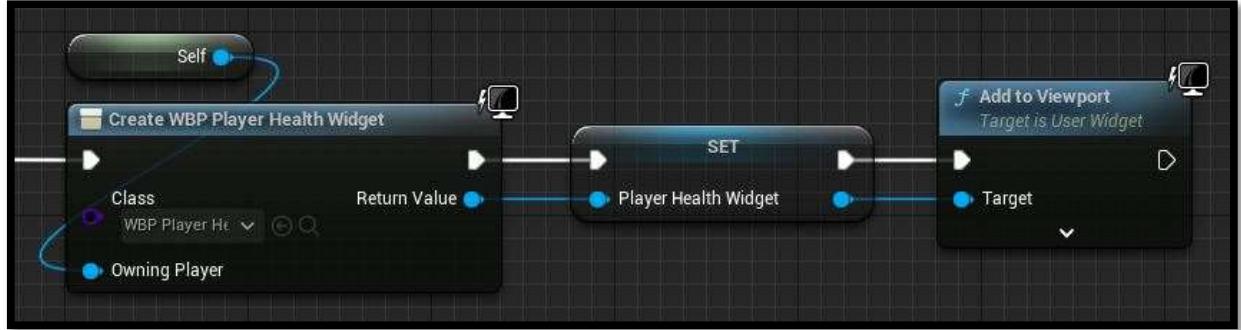
5. À partir du fil d'exécution **Then 0** du nœud **Sequence**, ajouter les nœuds de l'image ci-dessous :



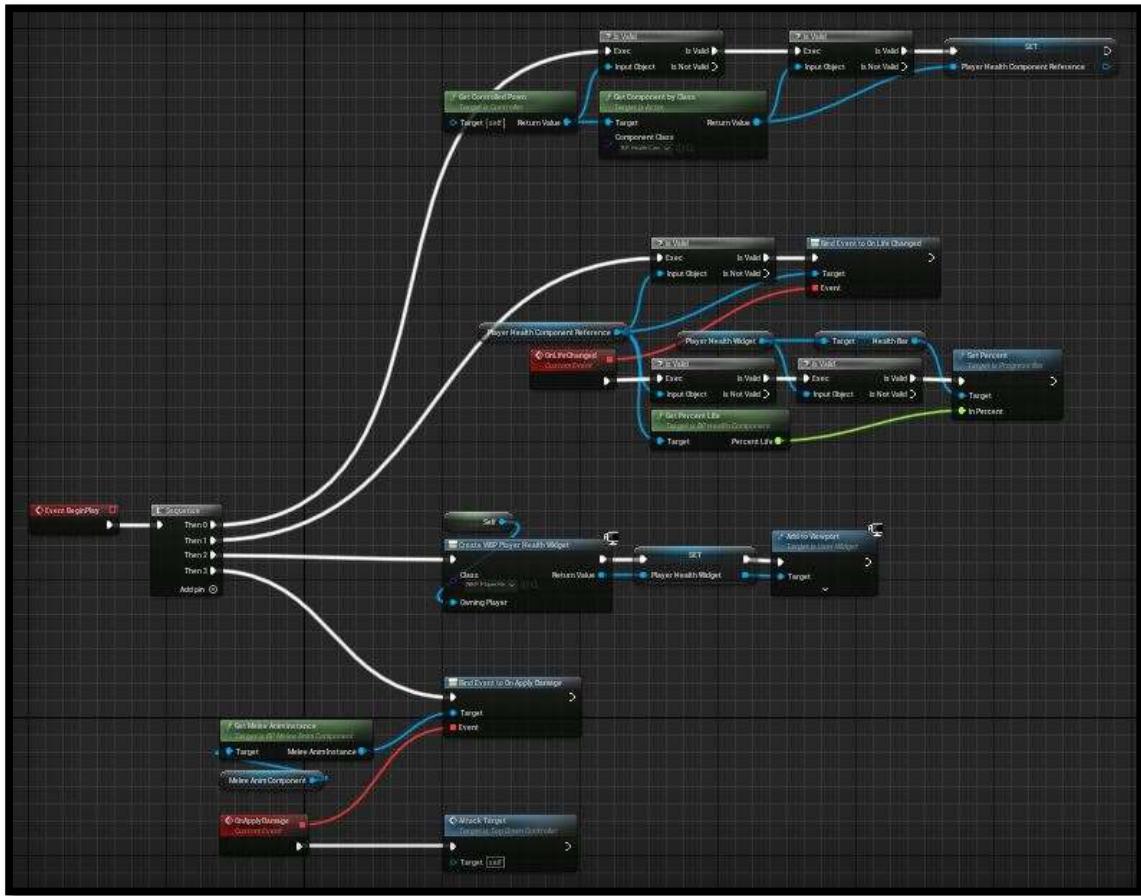
6. À partir du fil d'exécution **Then 1** du nœud **Sequence**, ajouter les nœuds de l'image ci-dessous :



7. À partir du fil d'exécution **Then 2** du nœud **Sequence**, ajouter les nœuds de l'image ci-dessous :



8. Vérifier que le graphe en fonction de l'événement **BeginPlay** ressemble à celui-ci :



9. Compiler, sauvegarder et fermer **TopDownController**

10. Lancer le jeu et vérifier que :

- la barre de vie apparaît au bas de l'écran
- lorsque l'ennemi endommage le joueur, la barre de vie est mise à jour

AJOUT D'EFFETS VISUELS ET SONORES (SI LE TEMPS LE PERMET)

AJOUT D'UN MENU PRINCIPAL (SI LE TEMPS LE PERMET)

POUR FINIR

Dans la section *Unreal Engine* de l'Epic Games Launcher, vous pouvez aller dans l'onglet *Learn* pour avoir accès à la documentation d'Unreal Engine 5 ainsi que plusieurs ressources pour vous apprendre comment utiliser l'engin au-delà de ce que nous avons vu aujourd'hui.

ANNEXES

ANNEXE 1 – INSTALLATION D’UNREAL ENGINE 5

Voici comment installer Unreal Engine 5 :

1. Ouvrez l'*Epic Games Launcher*
2. Allez dans la section *Unreal Engine*
3. Allez dans l’onglet *UE5*
4. Appuyez sur *DOWNLOAD EARLY ACCESS*
5. Suivez les étapes d’installation

ANNEXE 2 – AJOUT DE L’ASSET PACK *CITY OF BRASS : ENEMIES* AU PROJET

Voici comment ajouter City of Brass : Enemies à un projet :

1. Ouvrez le *Marketplace*
2. Dans la barre de recherche, écrivez : « City of Brass : Enemies »
3. Appuyez sur l’asset pack nommé *City of Brass : Enemies*
4. Appuyez sur *Free*, cela ajoutera l’asset pack dans votre librairie
5. Appuyez sur *Add To Project*, puis sélectionnez le projet dans lequel vous désirez ajouter l’asset pack
6. Appuyez sur *Add to Project*

ANNEXE 3 – RESSOURCES EXTERNES

- [Blueprint Class](#)
- [Blueprint Editor Reference](#)
- [Damage in UE4](#)
- [Behavior Tree Quick Start Guide](#)
- [Unreal Engine’s Asset Naming Convention](#)