République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou
Faculté de génie électrique et d'informatique
Département d'informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de
Master en Informatique

Spécialité : Systèmes Informatiques

Présenté par :
MERRAD Sofiane

Thème

Conception et réalisation d'un jeu sur Android avec le moteur de jeu Unreal

Soutenu le 24 novembre 2020 devant le jury composé de :

Mme. TAOURI Dalila	Présidente
M. SADI Samy	Examinateur
Mme. AMIROUCHE Fatiha	Directrice de mémoire

Dédicaces

A mes très chers parents qui se sont toujours sacrifiés pour me voir réussir.

A mes très chères sœurs « Kahina », « Samia » et « Farida ».

A mes précieux frères « Aziz » et « Youcef ».

Aux petits anges qui sont mes neveux « Hanane », « Aris » et « Rayane ».

A la mémoire de mes grands-parents et à tous ce qui ne sont plus là mais qui sont toujours présents de mon cœur.

A quatre personnes formidables qui ont toujours été là pour moi.

A mes aimables et honorables ami(e)s, collègues d'études.

Je vous dédie ce travail…

Remerciements

Tous d'abord, je tiens à remercier le bon dieu tout puissant de m'avoir donné la volanté et la force pour accomplir ce travail.

J'adresse ma plus haute gratitude à mon encadreuse **Madame Fatiha AMIROUCHE** de l'université Mouloud MAMMERI de Tizi Ouzou pour m'avoir encadré et orienté ainsi que pour son aide précieuse et surtout pour ses judicieux conseils.

Je tiens à remercier également les membres du jury d'avoir accepté d'évaluer mon travail.

Je voudrai remercier mes chers parents et toute ma famille pour leurs soutiens et leurs encouragements.

Enfin, je saisis cette occasion pour exprimer mes vifs remerciements à toutes les personnes qui ont contribué de près ou de loin à l'accomplissement de ce travail.

Table des matières

Table des figures

Chapitre I : Introduction à l'infographie

1. Introduction

L'infographie est une branche de l'informatique qui englobe les domaines de la **création** (ou **synthèse**) et de la manipulation d'images numériques assistée par ordinateur. Lors de l'introduction du concept dans la langue française vers les années 1970, le terme « infographie » désignait les graphismes produits par ordinateur. Puis, par confusion sur le sens du préfixe « info- » (informatique / information), le terme infographie fût utilisé par certains au sens de diagramme, cartographie et tout type de schéma explicatif destiné à mettre en image des informations, notamment statistiques ou géographiques. Mais le concept d'infographie désigne en français tous les graphismes produits par des moyens informatiques. L'infographie comprend aussi les techniques consistant à finaliser le travail du graphiste à l'aide de l'outil informatique : retouche photographique, mise en couleur de bandes dessinées, habillage de perspectives architecturales, etc. Ce métier est né avec l'avènement de l'informatique ; il est la continuité du graphisme sous toutes ses formes antérieures.

2. Historique

L'infographie est née au champ d'honneur de la recherche militaire : Côté militaire, c'est le secteur de la simulation de vol qui a contribué le plus à l'essor des images de synthèses. Les scientifiques ont commencé par simuler les vols à l'aide de films et de déplacement de caméra sur une maquette. Plus tard, la technologie a permis, à l'aide de l'ordinateur, d'afficher sur l'écran cathodique du pilote un paysage de synthèse correspondant aux déplacements réels de l'avion (figure I-1).

L'imagerie médicale a elle aussi favorisé le développement des images informatisées. Afin de mieux exploiter les éléments du diagnostic, les chercheurs, avec l'arrivée des fibres optiques, des ultrasons (on parle ici de l'échographie), des scanners et de la résonance magnétique nucléaire, ont utilisé grandement les bienfaits de l'infographie.

Le troisième secteur est celui de la recherche industrielle. On voit nettement une progression de l'utilisation de l'ordinateur dans ce domaine à des fins diverses telles que :

- Le DAO ou dessin assisté par ordinateur (la table à dessin informatisé).
- La CAO ou conception assistée par ordinateur.
- La FAO ou fabrication des objets assistée par ordinateur.
- La CFAO ou conception et fabrication assistée par ordinateur.

Figure I-1. Exemple de simulateur de vol.

- L'architecture rejoint la DAO et utilise aussi les images de synthèse, car elles donnent aux plans et aux projets de Construction une dimension photo réaliste. L'effet est bluffant et permet à la clientèle de visionner en avant-première son projet de Construction sur écran. Ce sont des « maquettes », crées sur écran, qui collent très fortement à la réalité, et qui offrent aux clients une qualité d'image haute définition.
- L'autre secteur qui a largement bénéficié des services de l'infographie est le secteur des créations artistiques. C'est peu à peu que l'infographie artistique s'est détachée des domaines scientifiques pour évoluer seule et devenir une technique autonome. A l'aube du travail artistique informatisé, il était plutôt complexe de créer des images via l'ordinateur. Ce travail nécessitait des connaissances en programmation informatique. L'artiste était toujours associé à un informaticien. Aujourd'hui, la convivialité des outils proposés par les fabricants de logiciels infographiques nous permet de nous exprimer librement et sans trop de contraintes. Dans le monde du cinéma la technologie 3D, et les images de synthèse sont utilisées, car elles permettent de créer des espaces, des images dont la seule limite est l'imagination. C'est le cas pour les effets spéciaux, et les films d'animation
- Enfin, le secteur des jeux vidéo est l'un des plus gros consommateurs d'infographie et d'images de synthèse. Nous consacrons à ce domaine, objet de notre travail, tout le chapitre 2.

3. La Synthèse D'image

3.1. Définition

La synthèse d'images consiste en la création par ordinateur, d'images virtuelles représentant des objets ou des scènes du monde réel ou imaginaire. Elle a pour objet la conversion d'une description symbolique d'une scène en une image d'un degré de réalisme proche de la photographie.

3.2. Problèmes de base de la synthèse d'images

Malgré les grandes différences entre les applications possibles de l'infographie, on retrouve toujours les mêmes problèmes de base à résoudre :

- La modélisation : C'est le passage de l'objet réel à sa représentation interne. La problématique fondamentale de la modélisation est de trouver la meilleure représentation possible pour les objets à modéliser.
- Le traitement (i.e. La manipulation d'objets) : Les objets modélisés sont stockés dans une structure de données (généralement une matrice de points), pour être manipulés soit directement par l'opérateur, soit par des logiciels, à travers des opérations géométriques comme l'union, l'intersection ou la différence d'objets existants pour en former de nouveaux, ou encore le déplacement (ou translation) d'objets, leur rotation, leur déformations…etc.
- La visualisation (ou affichage) : La visualisation a pour objectif de construire sur le dispositif de visualisation, une image représentant les objets modélisés. Sur les écrans matriciels, il s'agit d'allumer l'ensemble des points de l'écran correspondant aux points de la matrice de points qui représente l'objet à visualiser.
- Le rendu réaliste : Le but est de produire des images non discernables de photographies d'objets réels, en simulant à l'affichage des objets, les effets d'ombrage et d'illumination, de couleur, de texture et de transparence.

Nous nous intéressons dans ce qui suit à la modélisation des objets.

3.3. La modélisation

L'objectif de la modélisation est de représenter une scène réelle ou imaginaire par une structure de données transposable sur ordinateur. Or, il convient de dire que les objets réels sont extrêmement compliqués et il est quasiment impossible d'en donner une description exacte. Le rôle de la modélisation est alors de permettre une approximation aussi bonne que possible des objets réels. On peut classer les approches de modélisation selon la nature des objets à modéliser, en deux catégories : la modélisation des objets naturels, et la modélisation des objets géométriques [1].

A. Modélisation des objets naturels

Un objet naturel est un objet non créé par l'homme. Les plantes, les organes, les nuages, le feu, les montagnes, … sont des objets naturels. Il existe plusieurs approches pour la modélisation des objets naturels, dont :

a. La modélisation par les fractales :

Une figure **fractale** est un objet mathématique, telle une courbe ou une surface: chacune des parties de la figure est semblable à n'importe quelle autre. Les fractales sont générées par un processus itératif.

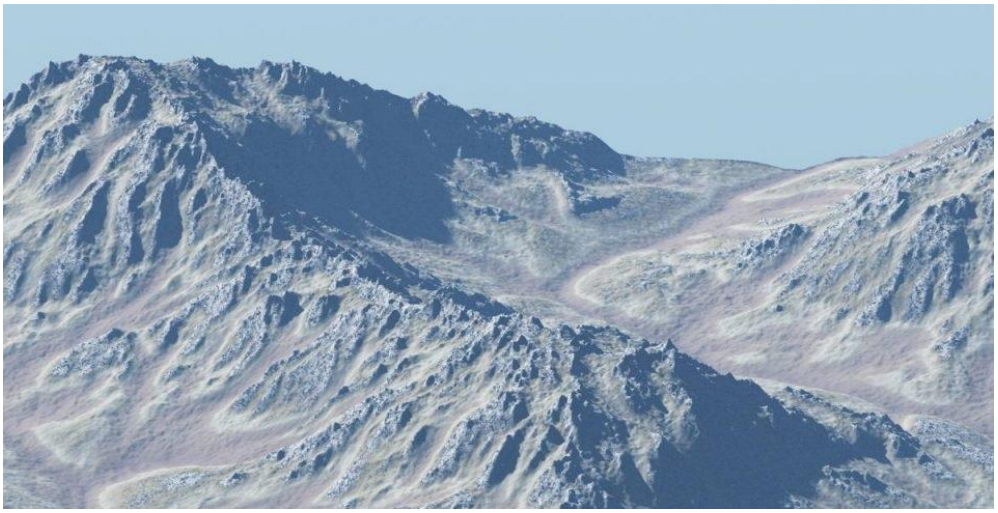


Figure I-2. Montagne fractale générée dans Eon Vue [2] (crédit Eon vue).

b. Modélisation par les graphtales

Les graphtales sont généralement utilisés pour modéliser des plantes et des arbres. Leurs formes sont déterminées par des règles d'une grammaire qui est interprétée de telle sorte qu'un élément de l'alphabet représente une partie élémentaire de l'objet (arbre ou plante).

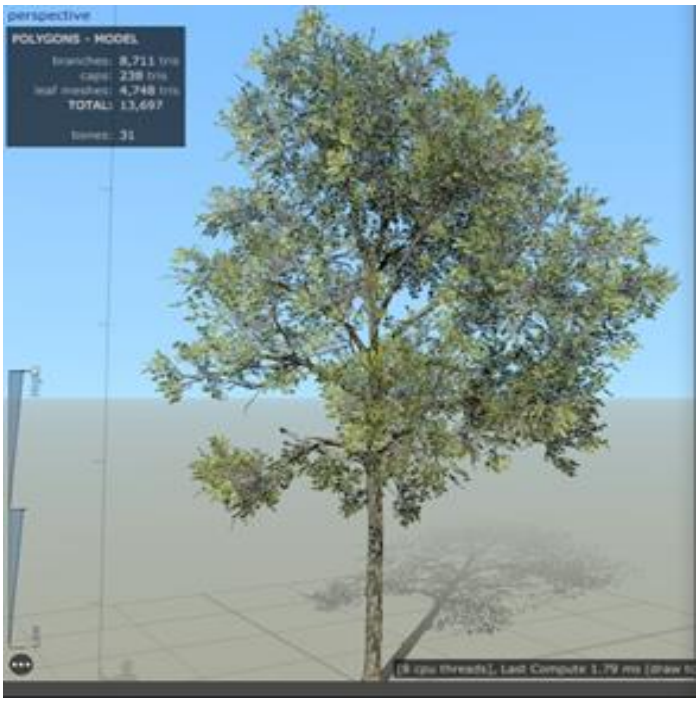


Figure I-3. Exemple de création de plante avec Speed Tree [3].

B. Modélisation des objets géométriques

Les objets géométriques sont des objets à aspects réguliers, généralement formés à partir de primitives géométriques de base (cube, sphère, pyramide, cône ou parallélogramme). Les objets créés par l'homme sont quasiment tous des objets géométriques.

Il existe plusieurs approches de modélisation des objets géométriques. La classification de ces approches est donnée en figure I-4 suivante. On s'intéressera dans la suite en particulier à la modélisation 3D (qui est la modélisation des objets en volume).

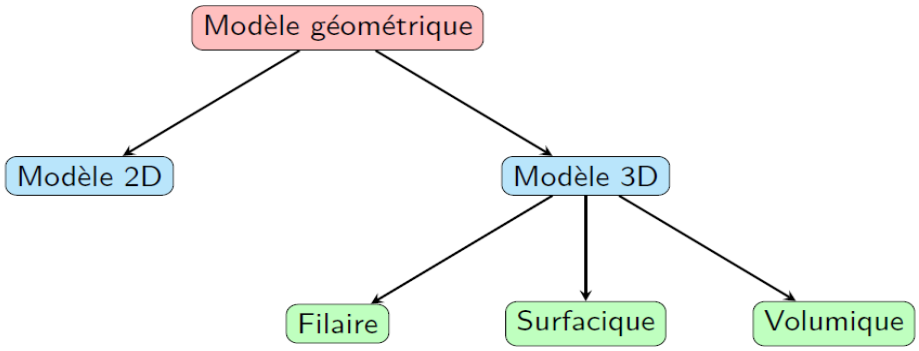


Figure I-4. Classification des approches de modélisation.

a. La modélisation Fil de fer

Dans la modélisation en fil de fer, l'objet est représenté par son squelette, c'est-à-dire par un ensemble d'arêtes et/ou d'arcs de cercles, lui donnant ainsi une apparence filaire (exemple de la théière en figure I-5).

Un des avantages de cette technique est une utilisation minimale d'espace mémoire et de temps de calcul ainsi qu'une visualisation rapide mais les inconvénients de cette technique résident dans son ambiguïté et de la possibilité de créer des objets sans aucun sens physique (figure I-6).

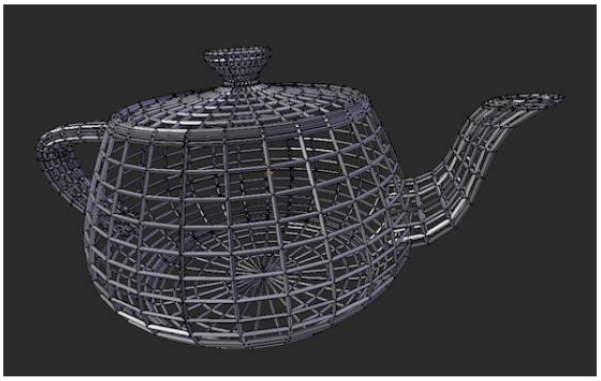


Figure I-5. Exemple d'un objet en fil de fer.

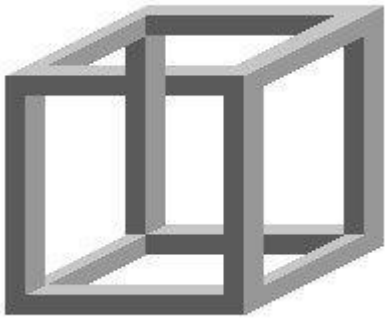


Figure I-6. Exemple d'un objet sans aucun sens physique.

b. La modélisation surfacique

Dans la modélisation surfacique tout objet en volume peut être défini par l'ensemble des surfaces le constituant, soit par une représentation par facettes planes ou une représentation par surfaces courbes.

Dans une représentation par facettes planes l'objet est défini par un ensemble de polygones. Cette approche n'est pas adaptée aux objets de formes courbes. En effet, dans ce cas, pour atteindre une bonne approximation de l'objet à représenter, les surfaces le constituant, doivent être assez nombreuses impliquant des temps de calcul plus élevés.

Figure I-7. Exemple d'utilisation des facettes plane dans la modélisation (crédit jeu Uncharted [4]).

- Une représentation par surfaces courbes consiste à approcher les surfaces d'un objet par un ensemble de familles de courbes connues telles que les courbes de Béziers et les B-splines.
  La représentation par surface courbes se base sur des courbes horizontales et des courbes verticales permettant de construire un maillage surfacique composé de patchs.
  La figure I.8(a) présente un exemple de surface de Bézier. La figure I.9(b) présente un exemple de surface B-spline.

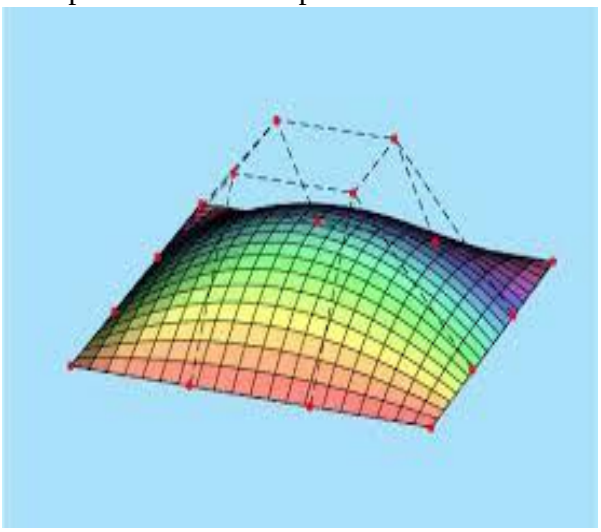
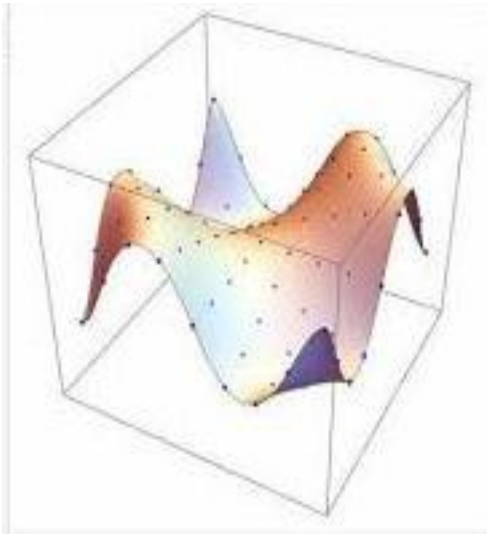


Figure I-8. (a) Surface de Bézier.



(b) Surface B-Spline

c. Modélisation volumique

La modélisation volumique est la modélisation la plus complète car elle englobe les deux types de modélisation précédemment décrites. Elle permet aussi une représentation dans l'espace d'un objet avec la notion de matière. Dans cette catégorie, nous citons les approches suivantes :

c.1. La représentation par extrusion

Cette représentation est basée sur l'animation d'un élément géométrique (généralement une arête ou une face), sur une trajectoire définie par une fonction d'animation (translation ou rotation), pour construire un objet 3D.

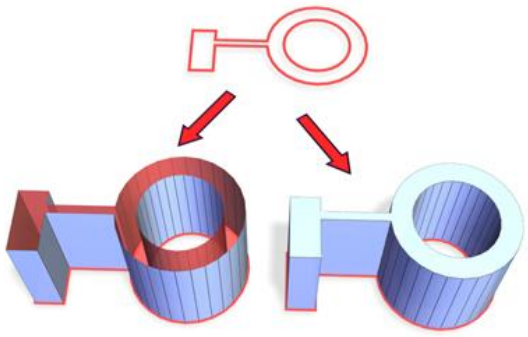


Figure I-9. Exemple de modélisation par extrusion.

c.2. Représentation par construction géométrique (ou modélisation CSG)

La représentation par construction géométrique considère que tout objet complexe peut être obtenu à partir de primitives géométriques de base (sphère, cylindre, cône, ...) en les combinant au moyen d'opérateurs ensemblistes (Union, Intersection, Différence).

Dans cette approche, l'objet à modéliser est représenté par un arbre dit arbre CSG, dont les nœuds sont les opérations ensemblistes binaires et les feuilles les objets primitifs ou des objets déjà calculées. La figure I-10 présente un exemple d'arbre CGG pour la modélisation d'une pièce mécanique.

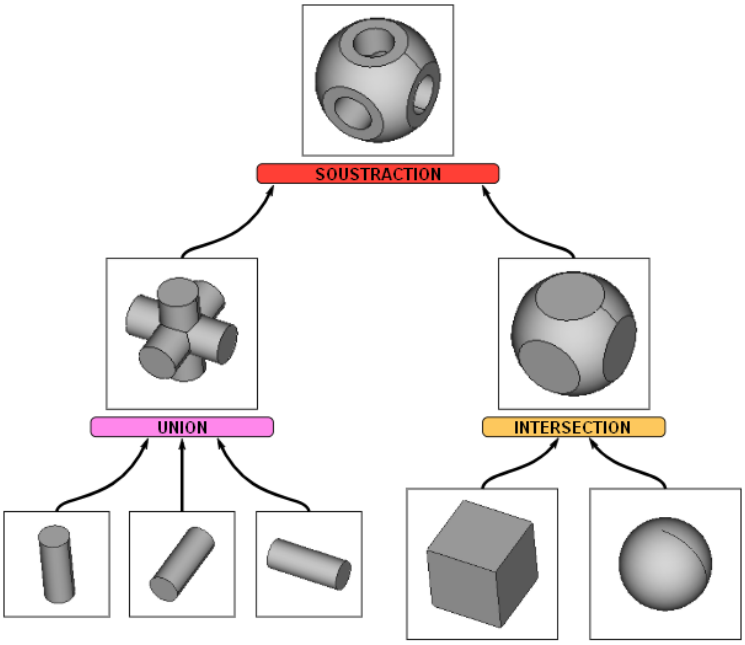


Figure I-10. Exemple d'arbre CSG.

Remarque : Il existe d'autres approches de modélisation hybrides, combinant plusieurs représentations pour modéliser une scène complexe, alliant tous les avantages des différentes méthodes tout en minimisant leurs points faibles.

4. Conclusion

Dans ce chapitre, nous avons introduit le domaine d'infographie, ses domaines d'application et nous avons aussi défini les différents problèmes liés à la synthèse d'image en nous intéressant de près à la modélisation. Dans le prochain chapitre nous allons étudier l'un des domaines qui utilise le plus l'infographie, à savoir les jeux vidéo.

Chapitre II : Les jeux vidéo

1. Histoire des jeux vidéo [5]

L'histoire du jeu commence vers la fin des années 1940, lorsque l'idée du jeu est née dans les universités de recherche en sciences informatiques. C'est en 1947 que les physiciens Thomas T. Goldsmith Jr. et Estle Ray Mann créent le premier prototype de jeu électronique baptisé le ***Cathode-ray tube amusement device,*** où le joueur utilise des tirs d'artillerie contre des cibles par l'intermédiaire de boutons et d'interrupteurs.

Le premier jeu sur ordinateur remonte quant à lui à 1950 avec la sortie du jeu et de la machine qui portent tous les deux la dénomination de ***Bertie the Brain.*** La machine qui pesait alors 1 tonne pour 4 mètres de hauteur permettait de jouer au jeu du morpion.

Ralph Baer inventeur germano-américain termine en 1968 le prototype de la toute première console de jeu vidéo qu'il nomme ***Brown Box*****. Magnavox** une société spécialisée dans l'électronique qui a embouché Ralph Baer sort en 1972 une version améliorée de la *Brown Box* qui sera commercialisé sous le nom ***Magnavox Odyssey,*** la console est vendue avec 12 jeux dont le célèbre ***Ping Pong***.

Durant les 1970 des centaines de consoles semblables à l'*Odyssey* sortent et les bornes d'arcade font leurs apparitions. La société Atari introduit sa borne d'arcade PONG 1 et vers la fin des années 1970 sa console l'ATARI 2600.

Durant les années 1980, les sociétés japonaises comme Nintendo, Sega ou Konami dominent le marché du jeu sortant des jeux célèbre comme Mario, Zelda, Pac-Man.

La fin des années 1980 est marquée par la sortie de plusieurs consoles portables telles que la GAME BOY, LYNX ou GAME GEAR.

La fin du vingtième siècle été marquée par une utilisation importante des ordinateurs et moindre pour les bornes d'arcade ainsi que la sortie de plusieurs consoles de maison où portable telle que la PlayStation 1.

Durant la décennie suivante, la société japonaise Sony proposera deux autres consoles qui sont la PlayStation 2 et la PlayStation 3, et la société américaine Microsoft entrera dans le marché avec ses consoles Xbox et Xbox 360.

Le marché du jeu sur ordinateur quant à lui a connu un énorme succès grâce aux jeux en ligne telle que les jeux de tir à la première personne (*fps* : *first person shooter*) comme ***Counter Strike*** et aux jeux de stratégie massivement multijoueur comme ***World Of Warcraf***.

Depuis l'apparition des systèmes d'exploitation IOS et ANDROID, l'utilisation des smartphones pour le gaming a pris une marge très importante dans l'industrie du jeu grâce à des jeux mythiques comme ***ANGRY BIRD*** où récemment ***FORTNITE***.

En 2019 le marché des jeux vidéo sur mobile pesait à lui tout seul 86 milliards de dollars de dépenses consommateurs dans le monde, contre 65 milliards de dollars pour le marché des consoles de maison et ordinateurs [6].

2. Réalisation d'un jeu vidéo

La réalisation d'un jeu passe par plusieurs étapes essentielles qui sont :

2.1. Le concept

Avoir l'idée du projet est la première étape dans la réalisation d'un jeu. Il est donc important de planifier le travail à effectuer pour la suite du projet.

Pendant cette étape, le concepteur de jeu (ou *game designer*) a constamment des idées sur le jeu qu'il veut développer, c'est pour cela qu'il résume toutes ses idées sous forme de textes ou de plans qui serviront ensuite à créer un cahier de charge du jeu.

Ernest W. Adams, un game designer et écrivain américain a décrit dans un exemple [7] qu'est disponible sur le site d'une école de génie électrique et informatique suédoise les informations nécessaires à la création d'un document de conception d'un jeu.

Le cahier des charges ou document de conception peut être simple où complexe dépendamment de la complexité du jeu décrit et comporte généralement les éléments suivants.

- **Le résumé du jeu :** Dans cette partie, le *Game designer* en charge de la conception du jeu va décrire l'idée principale du jeu, l'histoire, l'aspect artistique…etc.
- **La description du *gameplay* :** Le *gameplay* est l'ensemble des règles et des mécanismes qui vont créer l'expérience de jeu. Cette partie comportera un résumé de certains points du scénario tels que les objectifs à moyen et long terme à accomplir dans le jeu et les moyens (actions et outils) mis à la disposition du joueur pour y arriver.

2.2. Les recherches

Certains jeux vidéo ont besoin de recherches comme par exemple, sur un jeu historique qui se base sur un lieu précis, voire même sur une culture, des informations précises doivent être collectées afin de retranscrire l'effet dans le jeu.

Chaque objet important est sujet à des recherches afin d'obtenir une cohérence avec l'environnement et l'univers proposé par le jeu vidéo.

2.3. La sélection des outils

Avant de commencer la création du jeu, l'équipe de travail choisi les technologies à utiliser dans la création, l'exploitation et la maintenance du jeu tout au long de sa vie.

Parmi les outils indispensables à la création du jeu, il y a le **moteur de jeu** ou simplement un Framework de création de jeu. Ce moteur est un ensemble d'outils, de logiciels qui permettent de créer, visualiser, tester, et compiler le jeu pour différentes plateformes.

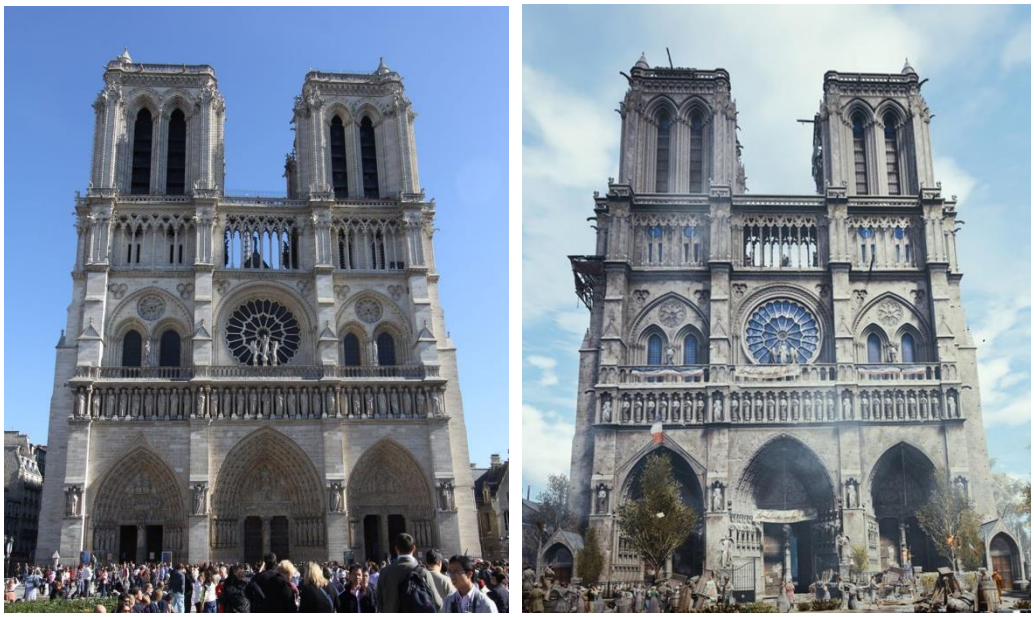


Figure II-1. Exemple de création d'environnement réel à partir de recherches.

En plus du moteur de jeu, il existe d'autres technologies, fréquemment utilisées dans des jeux en ligne, qui sont nécessaires à l'exploitation du jeu telles que des systèmes de connexions, de création et de sauvegarde de données sur un serveur.

2.4. La Création du jeu

La création du jeu comporte généralement deux aspects :

- Le *Level Art* ou création de mondes où évolueront les personnages du jeu, suivant les recherches effectuées. Dans un jeu, le monde peut être fermé ou ouvert. Dans un monde ouvert, les personnages peuvent se déplacer en toute liberté dans un monde « sans fin ». Dans un monde fermé, les personnages évoluent dans la même scène continuellement, sans évolution possible du monde.
- Le *Game Art* ou art du jeu consiste quant à lui à modéliser et intégrer tout ce qui n'est pas décor, comme par exemple les personnages (encore appelé caractères du jeu), les armes, les véhicules, l'intelligence artificielle, … etc. en se basant potentiellement sur les croquis et les diverses recherches effectuées en amont.

2.5. Les tests

Pendant toute la période de création d'un jeu, l'équipe procède à de multiples tests en interne pour repérer les éventuels bugs ou erreurs dans le jeu, ou simplement pour tester le gameplay du jeu.

Avant la publication d'une version finale du jeu, plusieurs versions de test appelées version Alpha ou Béta, sont testées par des testeurs de jeux vidéo, qui essayent de trouver tous les problèmes possibles et de donner leurs avis sur d'éventuels changements.

Figure II-2. Exemple de bug dû à une texture.

2.6. Commercialisation

Chaque société possède des buts à atteindre au niveau économique pour continuer à fonctionner, c'est pourquoi pour chaque jeu produit la société doit **rentabiliser** son investissement.
Selon la clientèle ou le type de plateforme de jeu visée, le système de monétisation du jeu diffère et voici un tableau non exhaustif des différentes façons de monétiser un jeu vidéo :

Modèles économiques	Exemples d'achats intégrés
Premium	Cosmétiques
DLC	Personnages
Free-to-play	Niveaux
Achats intégrés	Objets de gameplay
Publicités	Bonus

Figure II-3. Tableaux des différents modèles de monétisation d'un jeu vidéo.

Le modèle premium par exemple est une offre qui englobe toutes les fonctionnalités et DLC d'un jeu et qui sont commercialisées comme éditions spéciales souvent plus chère que la version standard.

Plusieurs combinaisons de monétisation peuvent être disponibles dans un seul jeu, exemple d'un jeu premium vendu dans des boutiques spécialisées, où une fois installé, le joueur a la possibilité

d'acheter de nouveaux contenus téléchargeables (DLC) et de souscrire à un abonnement pour jouer en ligne et de se procurer de nouveaux cosmétiques et personnages.

Les applications en général et les jeux en particulier sur mobile offrent un moyen de monétisation supplémentaire aux développeurs, il s'agit de la publicité. Les terminaux mobiles sont souvent connectés au réseau internet, de ce fait la mise en place d'un système de publicité est facilement réalisable grâce à des plateformes de publicité telles que Google Adsense [8].

Selon le marché visé par le jeu, des boutiques de commercialisation et de mise en vente de jeux sont nécessaires ou même obligatoire pour mettre en avant le produit. En général l'accès à ces boutiques est payant pour l'inscription ou la publication d'un jeu, et revendique un pourcentage des revenus générés par le jeu sur leurs plateformes.

2.7. Marketing

Dans un monde où des centaines de jeux sortent chaque mois sur différentes plateformes, la réussite d'un jeu ne se résume pas seulement à la création d'un bon produit, mais également à la visibilité de ce dernier.

L'enjeu majeur pour les sociétés, et surtout les développeurs freelances et les petites boites est d'avoir la plus large communication possible autour de leur jeu, d'où l'importance d'une bonne publicité au début, pendant la réalisation et essentiellement au lancement du jeu afin de garantir une vente optimale du produit.

3. Le moteur de jeu

3.1. Introduction [9]

Un moteur de jeu (*game engine)* est un ensemble de composants logiciels qui simule en temps réel le contenu d'un jeu ; il permet de se concentrer sur la création du jeu plutôt que de résoudre des soucis informatiques. Le moteur réalise les calculs indispensables et comprend plusieurs autres **moteurs** d'audio, graphique, physique.

Il existe une multitude de moteurs 2D/3D pour créer des jeux vidéo. La plupart permettent d'exporter sur PC, mobile et sur console. Certains moteurs sont totalement gratuits, d'autres offrent des versions gratuites avec des limites d'utilisation comme le cas pour **UnityPersonal**, d'autres encore sont payants via un abonnement comme c'est le cas pour **Unity Pro, o**u par redevance (*Royalty*) comme c'est le cas pour le moteur **Unreal Engine** (UE), où le développeur s'engage à verser 5% de ses revenus du jeu à la société propriétaire du moteur Unreal si ses revenus dépassent 1.000.000 de dollars.

3.2. Les fonctionnalités d'un moteur

Le moteur de jeu est une suite de logiciels et de moteurs qui traitent chacun une partie du développement du jeu. L'architecture diffère d'un moteur à un autre mais certaines fonctionnalités sont communes, dont :

a. Les Entrées/sorties

Cette partie s'occupe de la lecture des périphériques externes comme les joysticks, souris, claviers, et éventuellement des gestes tactiles. Elle permet aussi la lecture et la sauvegarde des données du jeu.

b. Le moteur de son

Le moteur de son combine un logiciel de lecture audio avec un logiciel de mixage et un générateur d'effets sonores (écho, compression, spatialisation). Les trois composants sont interconnectés.

Le moteur de son effectue en continu des calculs de synthèse sonore et de traitement numérique du signal. Certains moteurs de son simulent la réverbération, et le changement de timbre d'un son émis par une source éloignée, et produisent ainsi des illusions auditives.

c. Le moteur physique

Le moteur physique joue un rôle très important dans la création de jeu. Il permet de calculer le mouvement des objets, la manière dont ils interagissent les uns avec les autres, la manière dont ils glissent sur le sol ou sur les murs, la manière dont ils rebondissent…etc.

Il calcule aussi la déformation des objets mous, des cheveux, des vêtements et la destruction des objets solides. Il permet aussi de générer et de détecter les collisions d'objets qu'ils soient statiques ou dynamiques.

d. Le scriptage

Les langages de script sont souvent utilisés dans les jeux vidéo pour programmer le comportement des objets dynamiques. Ils sont aussi utilisés pour créer des interfaces utilisateurs, et aussi pour la création de matériaux pour les géométries.

Il existe deux types de langages couramment utilisés :

- *Les langages de script basiques* **:** Ce sont des langages basés, dans la majorité du temps, sur des langages de programmation comme le C++ ou C#. Leur utilisation nécessite une bonne connaissance du langage de toute l'équipe et un temps plus important pour la réalisation des tâches.
- *Les langages de script visuel* **:** Les langages de script visuel sont des langages dits langages de société. Basés sur des langages de programmations traditionnels, ils permettent à un designer ou un scénariste de configurer d'une manière visuelle, sans avoir à taper du code, tous les aspects du jeu. A titre d'exemple, le langage *Blueprint* permet le scriptage visuel dans le moteur de jeu *Unreal Engine*.

e. Intelligence artificielle

L'intelligence artificielle est une partie très importante dans les moteurs de jeu actuels, elle permet de rendre le monde dépeint par le jeu le plus cohérent et le plus humain possible afin d'améliorer le plaisir et l'immersion du joueur.

L'AI permet d'améliorer le comportement des personnages non joueurs ou encore générer du contenu de façon automatique.

f. Graphisme 2D/3D

Un moteur de jeu embarque toujours un moteur graphique 2D/3D qui permet de visualiser en temps réel le développement du jeu. Il permet aussi de visualiser les rendus des jeux dans plusieurs versions, utilisant pour cela les différentes bibliothèques graphiques (API) destinées aux ordinateurs, aux mobiles et même aux consoles, telles que OpenGL, DirectX, Vulkan, OpenGL ES.

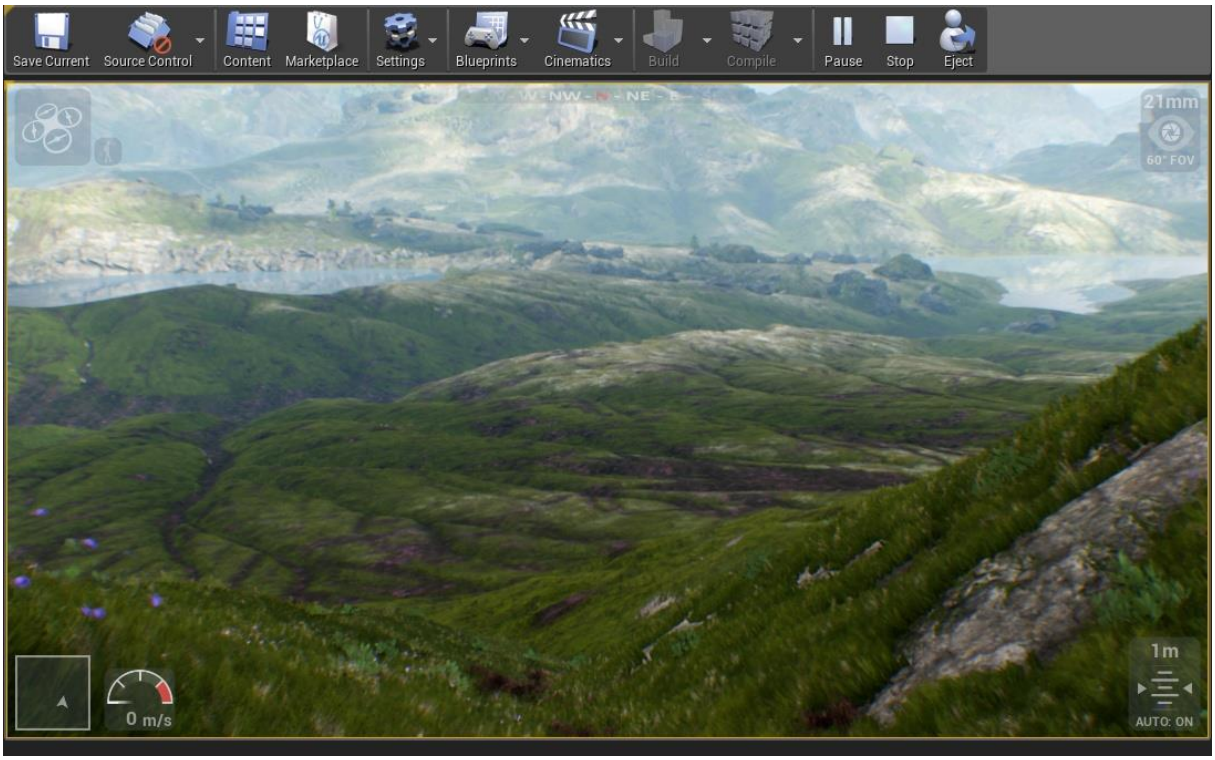


Figure II-4. Exemple de rendu en temps réel d'une scène dans le moteur Unreal Engine.

4. Conclusion

Dans ce chapitre nous avons résumé brièvement l'histoire des jeux vidéo, puis nous avons défini les étapes de réalisation d'un jeu vidéo, et enfin nous avons introduit les fonctionnalités d'un moteur de jeu. Dans le prochain chapitre nous allons décrire la conception de notre jeu EGGS ADVENTURE en détaillant son cahier de conception.

Chapitre III : Conception du jeu EGGS ADVENTURE

1. Introduction

La conception d'un jeu a principalement comme objectif d'une part d'avoir une idée précise et solide du jeu qui sera créé, la manière dont il sera joué, les mécaniques récurrentes, du cœur de gameplay et de son scénario, et d'autre part de poser les bases de l'architecture logicielle sur laquelle il va s'articuler.

Dans ce chapitre nous allons introduire la conception de notre jeu EGGS ADVENTURE.

2. Synopsis du jeu

EGGS ADVENTURE est un jeu d'arcade à la troisième personne dans lequel le joueur incarnera Mayk un jeune lapin qui pendant ses vacances sur son île lointaine, reçoit une lettre d'un membre de sa famille, lui demandant de les aider.

Mayk, le héros utilise une arme qui a une forme de marteau pour vaincre ses ennemis et peut aussi utiliser des bombes qu'il ramasse dans les différents niveaux pour faire subir des dégâts à ses ennemis.

Le monde du héros se compose de trois lieux totalement différents qui se situent dans des époques différentes : la période contemporaine, l'âge d'or de la piraterie, et le moyen âge.

Le héros gagne des pièces d'or en vainquent les ennemis qui se trouvent dans les différents niveaux. Ces pièces pourront être utilisées pour améliorer la santé du personnage et les dégâts infligés par l'arme de ce dernier, ou pour modifier son apparence en utilisant différents T-shirts, shorts, chapeaux et chaussures.

Des œufs sont souvent disponibles dans les niveaux permettant au joueur de gagner soit des :

- **Bombes :**

Les bombes infligent plus de dégâts que l'arme principal, ces bombes peuvent être ramassées en s'approchant d'un ŒUF BOMBE.

On prévoit 4 types de bombes différentes les unes des autres par leurs couleurs et leurs points de dégâts :

- Bombes vertes qui infligent 100 points de dégât.
- Bombes rouges qui infligent 75 points de dégât.
- Bombes blues qui infligent 50 points de dégât.
- Bombes jaunes qui infligent 25 points de dégât.

- **Bonus :**

Pendant une partie le joueur peut ramasser des ŒUFS BONUS qui lui offriront :

- Des bonus pendant le jeu : c'est des bonus que le joueur utilise immédiatement dans le jeu lui offrant ainsi des secondes de jeux supplémentaires, des pièces d'or ou des points de vie.
- Des bonus débloquant des améliorations : ce sont des bonus qui débloquent des tenues supplémentaires pour le joueur, ou lui offrent des pièces d'or.

Pour chaque niveau, le héros doit vaincre un nombre précis d'ennemis pour pouvoir à la fin libérer son peuple.

3. Document de conception du jeu EGGS ADVENTURE

3.1. Nom du jeu

-EGGS ADVENTURE.

3.2. Nom du Héros

-Mayk Darrem.

3.3. Motivation du joueur

Le joueur s'est vu confié la mission de sauver le pays de Mayk en combattant ses ennemis. Pour chaque ennemi vaincu il gagne des pièces d'or qui lui permettent de débloquer des personnalisations supplémentaires et d'améliorer son arme et sa santé dans le jeu.

3.4. Caractéristiques

- Un jeu totalement en 3D inspiré de jeux d'arcade et d'aventure avec des personnages détaillés et personnalisables.
- Un temps de jeu infini avec des sessions de 150 secondes pour chaque partie.
- Des dizaines d'améliorations et de tenues à débloquer et à gagner.
- Un jeu destiné à toutes les tranches d'âge.
- Des paramètres ajustables aux différents téléphones ou tablettes.
- Des ennemis dotés des dernières avancées en intelligence artificielle.
- Des bonus cachés à trouver dans les parties.
- 03 univers totalement différents modélisés en 3D.

3.5. Plateforme ciblée

Le Jeu sera disponible en premier lieu pour Android sur le Google Play et pourra être porté sur d'autres plateformes comme I'IOS, Windows, Mac.

3.6. L'histoire du jeu

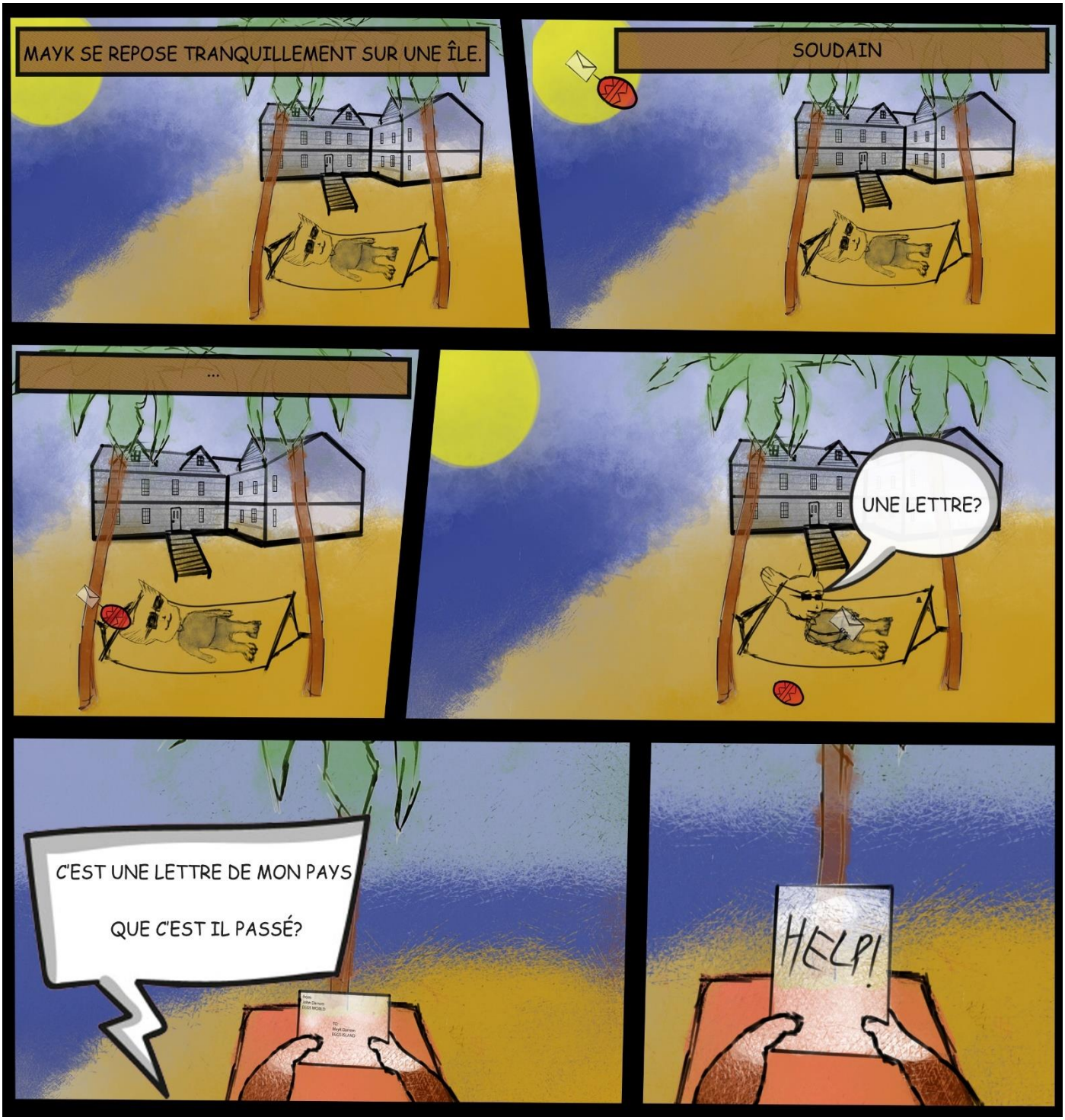


Figure III-1. Concept art* de l'histoire du jeu.

* Le Concept art est une forme d'illustration utilisée pour traduire une idée.

3.7. Les contrôles et les interactions

Le joueur peut interagir avec le caractère via des mécanismes et actions. La liste des actions que le joueur peut exécuter sont :

- Le joueur peut se déplacer dans l'univers.
- Le joueur peut sauter.
- Le joueur peut attaquer avec son arme.
- Le joueur peut jeter des bombes sur les ennemis.

3.8. Conception graphique : modélisation des éléments du jeu

Le jeu EGGS ADVENTURE est principalement constitué des objets suivants :

- Les personnages, Mayk et les ennemis.
- Les armes et œufs.
- Autres éléments des niveaux : maisons, arbres, bornes à incendie…etc.

3.8.1. Les personnages

- **Mayk** est le personnage jouable par l'utilisateur. Son modèle graphique a été téléchargée du site Mixamo [10].
  - La modélisation est basée sur des surfaces courbes qui sont obtenues grâce à des formes de base comme des sphères et des cylindres.
  - La figure suivante représente les étapes de la création du personnage, les étapes une et deux représentent la création du modèle avec des sphères et des cylindres. La modélisation du personnage est ensuite affinée dans les étapes trois et quatre puis texturé en étape cinq pour obtenir l'aspect final du caractère dans l'étape six.

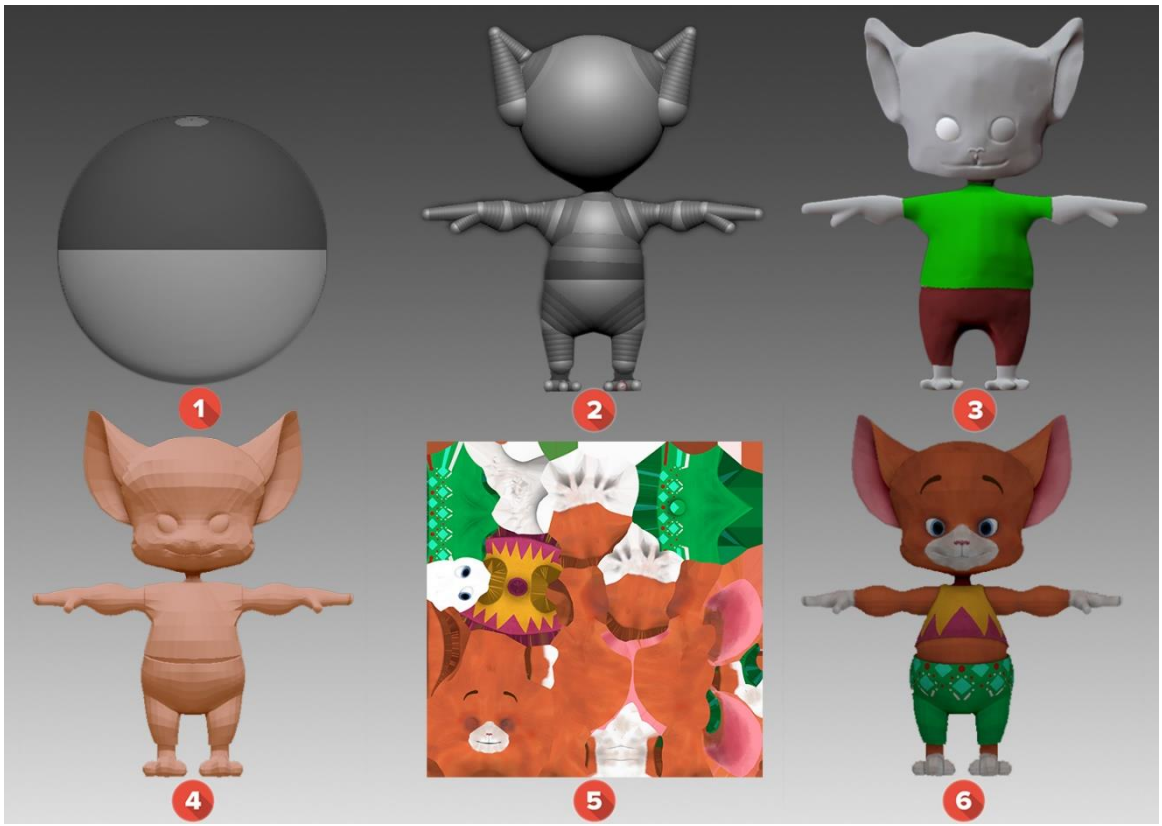


Figure III-2. Etapes de modélisation du personnage Mayk.

- Les objets de personnalisation comme les T-shirts et Shorts et les chapeaux sont obtenus grâce à des modélisations par extrusion des parties du corps du personnage.

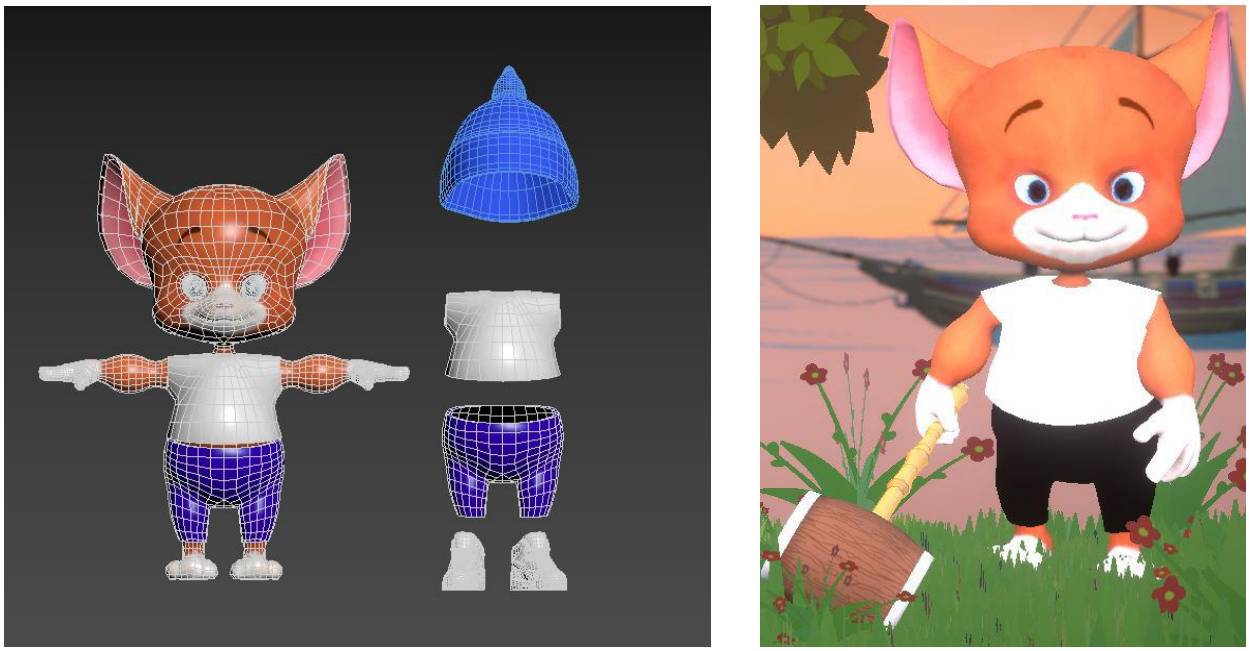


Figure III-3. Modélisation et rendu réaliste du personnage Mayk et ses différentes personnalisations.

- **L'ennemi** est une variante du personnage Mayk, sa modélisation de base est la même avec celle du personnage Mayk avec des modifications qui sont :

  - Une modification des traits du visage comme les sourcils et les lèvres représentée par l'étape 2 de la figure III-4.
  - L'ajout d'une tenue noire obtenue grâce à une modélisation surfacique et par extrusion du corps du personnage représentée par l'étape 3 de la figure III-4.

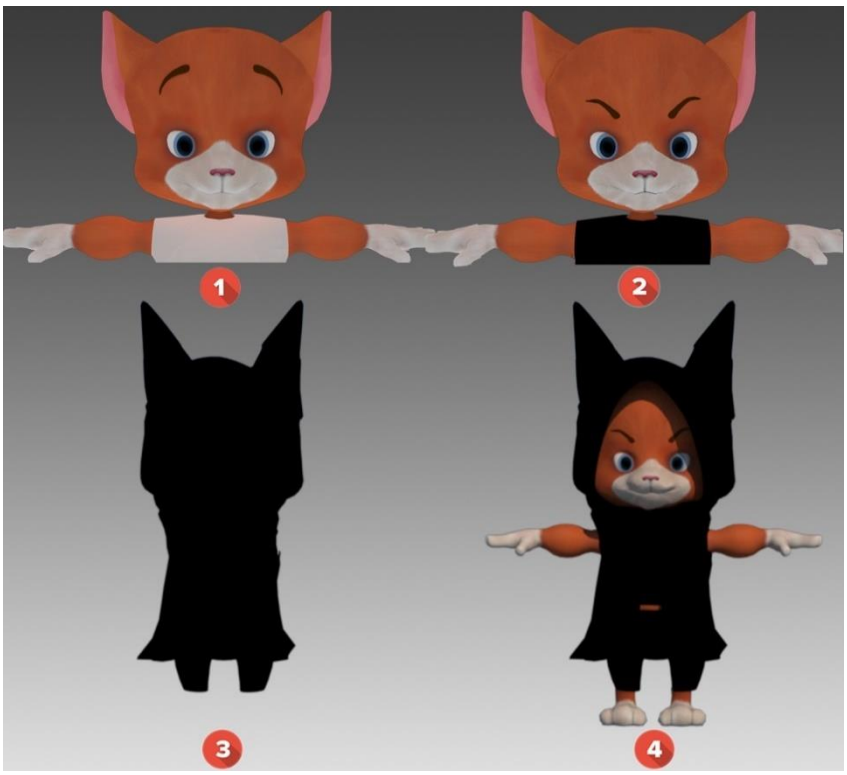


Figure III-4. Etapes de création du personnage ennemi.

- Ce personnage est autonome et indépendant du joueur, et il est conçu pour être géré automatiquement par le système du jeu. On lui prévoit des capacités de combat ainsi que des habilitées spécifiques.

Figure III-5. Modélisation finale et rendu réaliste du personnage ennemi.

3.8.2. Les armes et œufs

a) Armes

Les personnages dans EGGS ADVENTURE disposent tous d'une arme principale qui peut être utilisée pour vaincre un ennemi proche.

Cette arme inflige des points de dégât à chaque contact avec un ennemi, ces points de dégât peuvent être améliorés dans le menu de personnalisation dans le cas du héros.

Figure III-6. Prototype de l'arme principale des personnages.

La modélisation de l'arme se base sur l'union de deux parties :

- Le bras de l'arme : Ce dernier est créé grâce une modélisation par extrusion d'un cercle selon un axe. Une déformation de cette extrusion est faite tous le long de l'axe d'extrusion en changent le diamètre du cercle.
- La tête de l'arme est une modélisation par CSG en soustraient deux sphères aux extrémités d'un cylindre plein.

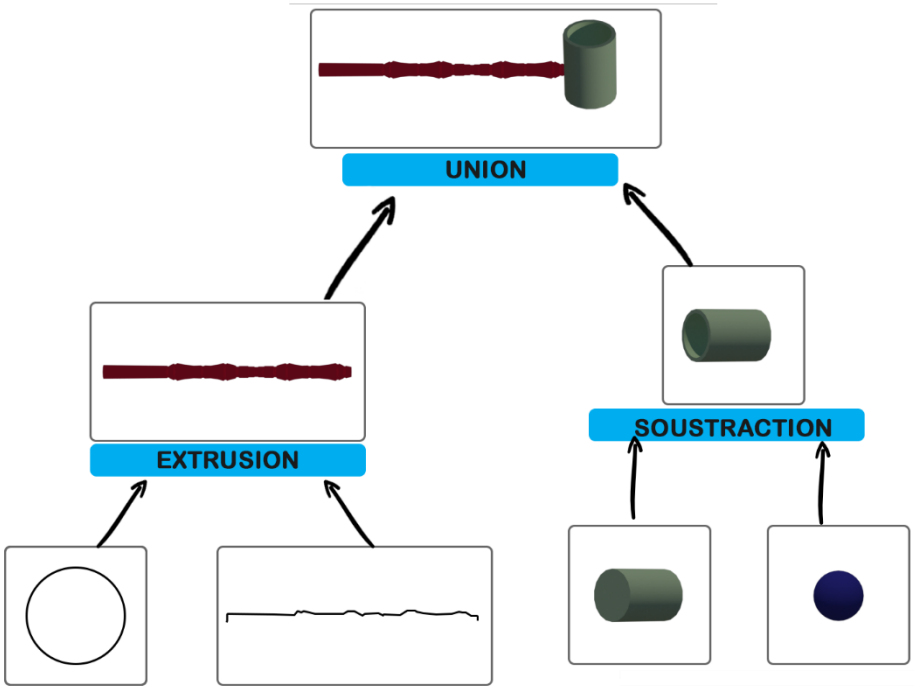


Figure III-7. Etapes de la création de l'arme principale.

b) Œufs

Les œufs ont pour but dans le jeu EGGS ADVENTURE, d'équiper le personnage de bombes qu'il pourra utiliser pour infliger des dégâts aux ennemis, ainsi que de lui offrir des bonus utilisables dans la partie ou des bonus qui offrent des pièces d'or et des personnalisations.

- La modélisation d'un œuf pour le jeu EGGS ADVENTURE est une modélisation par surface courbes réalisée grâce à une déformation elliptique d'une sphère.

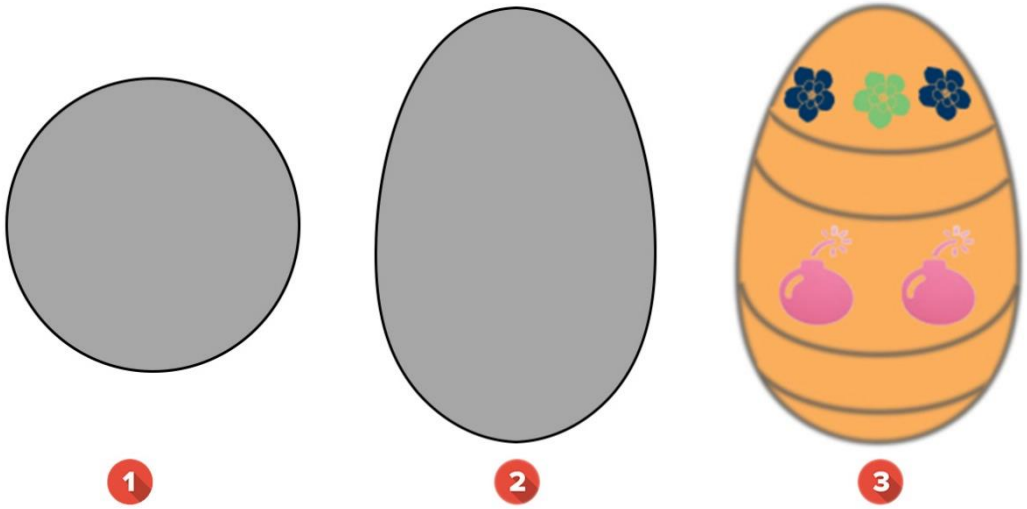


Figure III-8. Prototype de création d'un œuf.

3.8.3. Les niveaux

Les différents éléments qui composent les scènes du jeu sont issus d'assets* et de modèles disponibles dans la boutique d'Unreal, ainsi que de modélisations personnelles.

- Les objets tels que les habitations sont issus d'une combinaison de modélisation par facette plane et de modélisation par extrusion.
- L'étape 2 de la figure III-9 est issue d'une découpe d'un cube selon trois lignes de découpage. A chaque découpage, la face obtenue en haut est extrudée pour créer la forme du toit.
- L'étape 3 de la figure III-9 consiste à rajouter les éléments secondaires de la maison comme les portes et fenêtres.

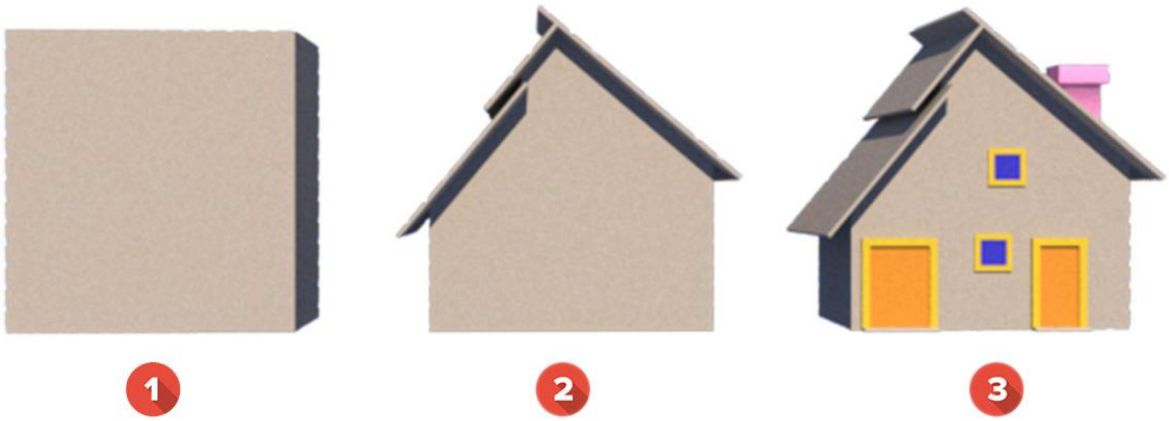


Figure III-9. Etapes de création d'une habitation pour le jeu EGGS ADVENTURE.

- Les objets secondaires comme les bornes à incendie ou les caisses sont aussi issus d'une combinaison de modélisation par facette plane et de modélisation par extrusion.
- L'étape 1 de la figure III-10 représente l'extrusion d'un hexagone selon un axe, tout en changeant le diamètre du polygone pendant l'extrusion pour avoir l'étape 2.
- L'étape 3 consiste à rajouter des éléments secondaires comme les robinets d'eau.
- Les arbres sont modélisés dans des logiciels spécialisés à l'aide de graphtales.

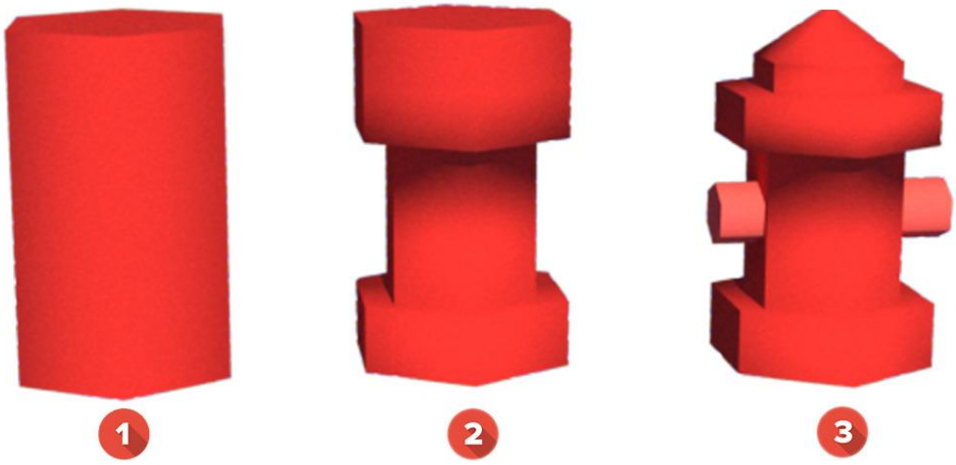


Figure III-10. Etapes de création d'un objet secondaire pour le jeu EGGS ADVENTURE.

*Tous les éléments externes qui peuvent être nécessaires à une scène.

- L’étape 1 de la figure III-11 consiste à la création du tronc d’arbre.
- L’étape 2 de la figure III-11 consiste à rajouter des branches au tronc.
- L’étape 3 de la figure III-11 consiste à rajouter des feuilles aux branches.

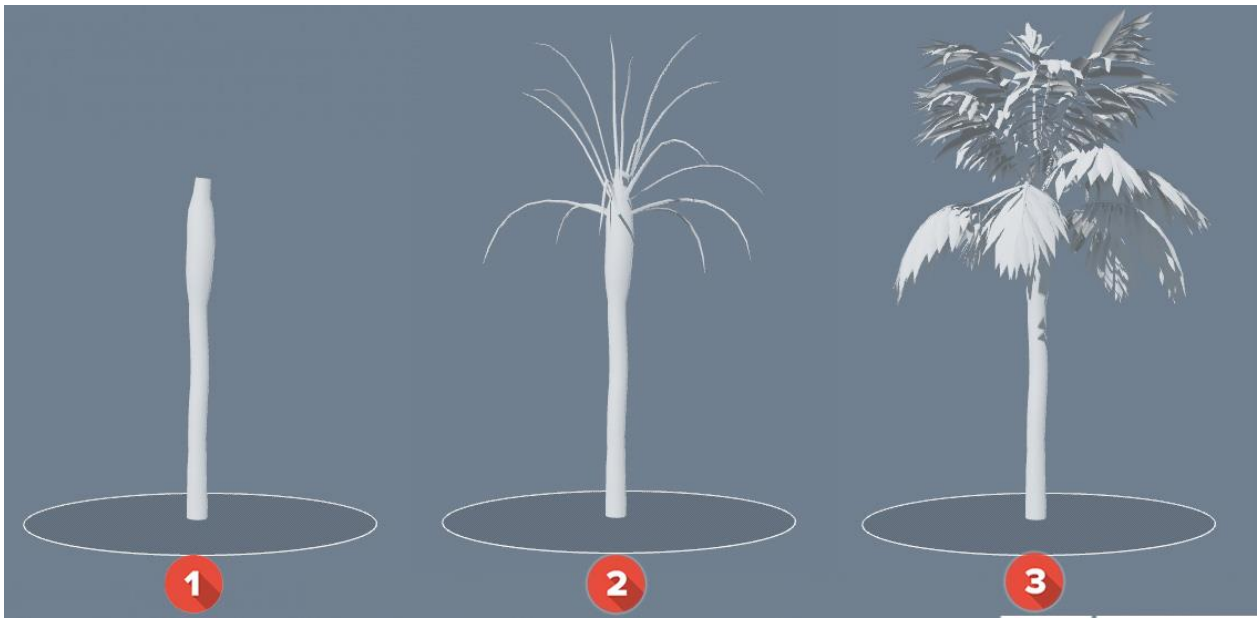


Figure III-11. Etape de création d’un arbre pour le jeu EGGS ADVENTURE.

Chaque niveau contient des centaines d’instances de ces objets. Ils existent quelques méthodes pour éviter d’avoir tous les objets identiques dans une scène comme :

- Une mise en échelle et une rotation aléatoire est appliquée aux objets secondaires.
- Une application de textures différentes pour différencier les habitations.
- Création d’habitations avec différente hauteurs.
- Séparation de deux instances d’un même objet par au minimum une instance d’un autre objet.

Niveau I : période contemporaine

Le premier niveau comporte des éléments de la période contemporaine tels que des maisons en ciments, des routes en bitume, des lampadaires et bornes d'incendie.

Figure III-12. Exemple de quelques éléments du niveau 1.

Figure III-13. Capture d'écran du niveau 1.

Niveau II : Age d'or de la piraterie

Le deuxième niveau se situe entre le 17ème et 18ème siècle pendant la période de la piraterie. Le niveau est situé sur une île dont les éléments les plus représentés sont des maisonnettes en bois, des palmiers, et des bateaux.

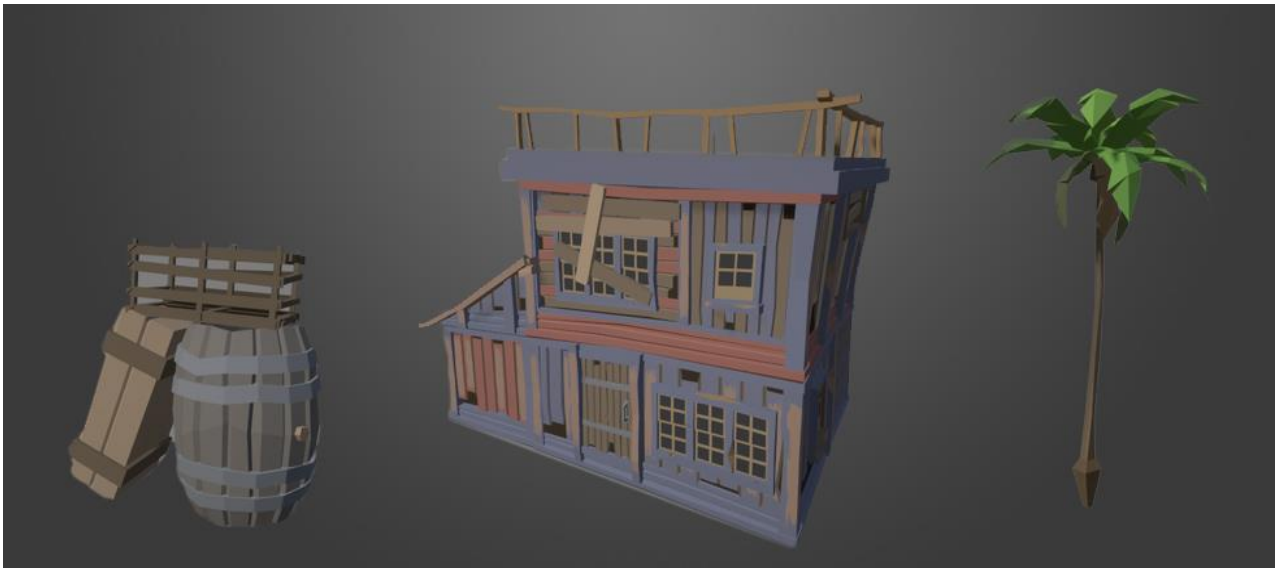


Figure III-14. Exemple de quelques éléments du niveau 2.

Figure III-15. Capture d'écran du niveau 2.

Niveau III : moyen âge

Le troisième niveau prend comme époque de référence le moyen âge. Le niveau est formé d'une forteresse dont laquelle se trouve plusieurs habitations, routes pavées et de châteaux.

Figure III-16. Exemple de quelques éléments du niveau 3.

Figure III-17. Capture d'écran du niveau 3.

4. Conception informatique

Dans cette partie, nous allons concevoir la hiérarchie des différentes classes du jeu qui représentent les différents objets utilisés dans la création du jeu EGGS ADVENTURE. Nous allons ensuite définir et présenter les diagrammes de classe de quelques objets du jeu ainsi que les différents diagrammes de cas d'utilisation et de séquence des interfaces qui forment le menu principal.

4.1. Les types d'objets manipulés dans le jeu EGGS ADVENURE

Notre jeu se compose de plusieurs types d'objets, qui peuvent être classées en 3 catégories :

1. **Objets statiques :** Ce sont des objets qui composent l'environnement du jeu, ils n'ont aucune interaction avec le joueur. Ces objets sont souvent issus de logiciels de modélisation 3D ou de logiciels 2D et sont placés dans les différents niveaux, à l'exemple des maisons, arbres et bornes à incendie.

Figure III-18. Exemple d'environnement composé d'objets statique.

2. **Objets dynamiques :** ce sont tous les éléments qui ne font pas partie de l'environnement et qui accomplissent ou subissent des actions préalablement configurées. Ces objets sont souvent composés d'une partie visible qui est l'objet en soit et d'une partie implémentée qui est le code, qui définit les actions de cet objet.

Figure III-19. Exemple d'objets dynamiques formant une image.

3. **Autres objets définissant la logique du jeu**

   Dans notre jeu, on utilise en outre des objets invisibles dans les scènes du jeu, qui définissent la logique du jeu à l'exemple des *Game Mode*.

4.2. Conception informatique du jeu EGGS ADVENTURE dans Unreal

Tous les objets utilisés dans le jeu EGGS ADVENTURE sont classifiés selon une classification établie par le moteur Unreal. Dans cette classification, les objets dynamiques sont représentés avec plusieurs classes, chaque classe dispose de ses propres caractéristiques. Toute ces classes sont étendues d'une classe mère appelé '*Object*'. La figure III-20 représente la hiérarchie des classes dynamiques qui définissent notre jeu. Cette Hiérarchie est obtenue par extension d'une hiérarchie de base proposée par le moteur de jeu Unreal.

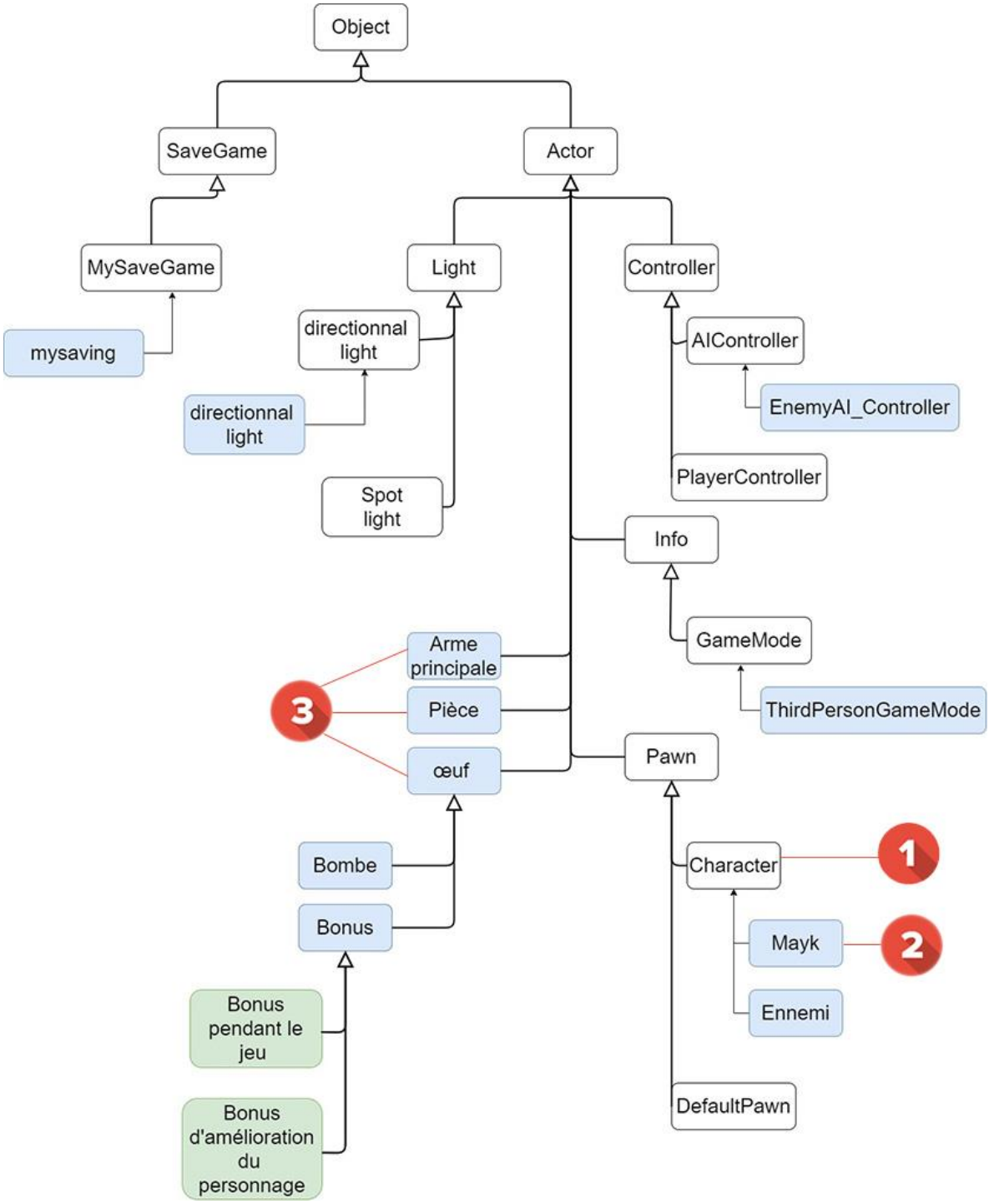


Figure III-20. Hiérarchie de quelques classes importantes dans le jeu EGGS ADVENTURE

La classe la plus importante est la classe ***'Actor'***, cette classe représente tous ce qui peut être placé, ou engendré par le moteur dans une partie et qui est contrôlé par l'utilisateur ou le moteur lui-même. Les éléments de la classe *'Actor'* jouent des rôles dans la phase de jeu et plusieurs autres classes héritent de cette dernière :

1. **Classe Pawn :** Tous les éléments jouables par l'utilisateur ou l'AI, héritent de cette classe.
2. **Classe character :** Classe fille de la classe Pawn, elle englobe tous les caractères ayant un mesh, un système de collision et une logique de mouvement pouvant être joués par le joueur ou l'AI.
    a. **Mayk :** C'est le caractère jouable par l'utilisateur.
    b. **Ennemi :** C'est le caractère qui représente les ennemis du Mayk.
3. **Classe AIController et PlayerController :** Ce sont des classes non physique composées seulement de scripts, appelées respectivement par les caractères gérés par l'AI et le joueur pour exécuter certaines tâches avec des fonctions particulières comme *'EventPossess'* et *'Event Unpossess' dès* que L'AI ou le joueur prennent possession de leurs caractères ou perdent le control de ces derniers.
    a. EnemyAI_Controller : C'est le contrôleur utilisé dans le jeu EGGS ADVENTURE pour permettre aux ennemis d'utiliser leurs arbres de décisions.
4. **Classe Light :** Comme son nom indique cette classe qui hérite de la classe Actor, elle gère toutes les lumières disponibles dans le moteur.
    a. Directionnal light : Elle représente le système de lumière utilisé dans les différents niveaux pour simuler la lumière du soleil.
5. **Classe Game Mode :** Cette classe hérite de la classe *'info'*. Toutes les classes de cette catégorie ne sont pas physique et servent en premier lieu aux échanges, communication d'informations pendant les parties de jeu. La classe Game Mode est utilisé fréquemment pour définir les règles de la partie, gérer les scores des joueurs, et gérer les actions des joueurs. L'appel à une instance de cette classe se fait automatiquement lors de la création d'un niveau de jeu.
    a. ThirdPersonGameMode : C'est une instance de la classe Game Mode utilisé dans le jeu.
6. **Classe Bonus :** C'est la classe qui représente tous les bonus disponibles dans le jeu, elle dispose d'un mesh sous forme d'un œuf et d'une partie script qui aléatoirement offre des bonus au joueur.
7. **Classe Bombe :** C'est la classe qui représente les bombes que le joueur peut utiliser pour infliger des dégâts aux ennemis. Elle dispose aussi d'un mesh sous forme d'un œuf.
8. **Classe Pièces :** Elle représente les pièces d'or que le joueur gagne en vainquant des ennemis. Cette classe dispose d'un mesh qui une forme d'une pièce et d'une partie script qui incrémente à chaque exécution le nombre de pièces gagnées par le joueur.
9. **Classe Arme prinicpale :** C'est la classe qui représente l'arme principale utilisée par le caractère Mayk et l'ennemi pour infliger des dégâts.

La classe ***'SaveGame'*** hérite directement de la classe *'Object'*, elle permet de créer des sauvegardes persistantes sur le support de stockage de l'appareil. La classe **Mysaving** est l'instance de la classe SaveGame dans EGGS ADVENTURE et stocke des informations telles que le nombre de pièces gagnées, les améliorations débloquées et le nombre d'ennemis vaincus.

Les éléments statiques comme les vêtements du personnage Mayk n'héritent pas de la classe Actor, et peuvent être rattachés à des éléments dynamiques comme les caractères (voir figure III-22).

La figure suivante détaille la partie de la hiérarchie définissant les caractères du jeu notée avec 1 précédemment vu dans la figure III-20

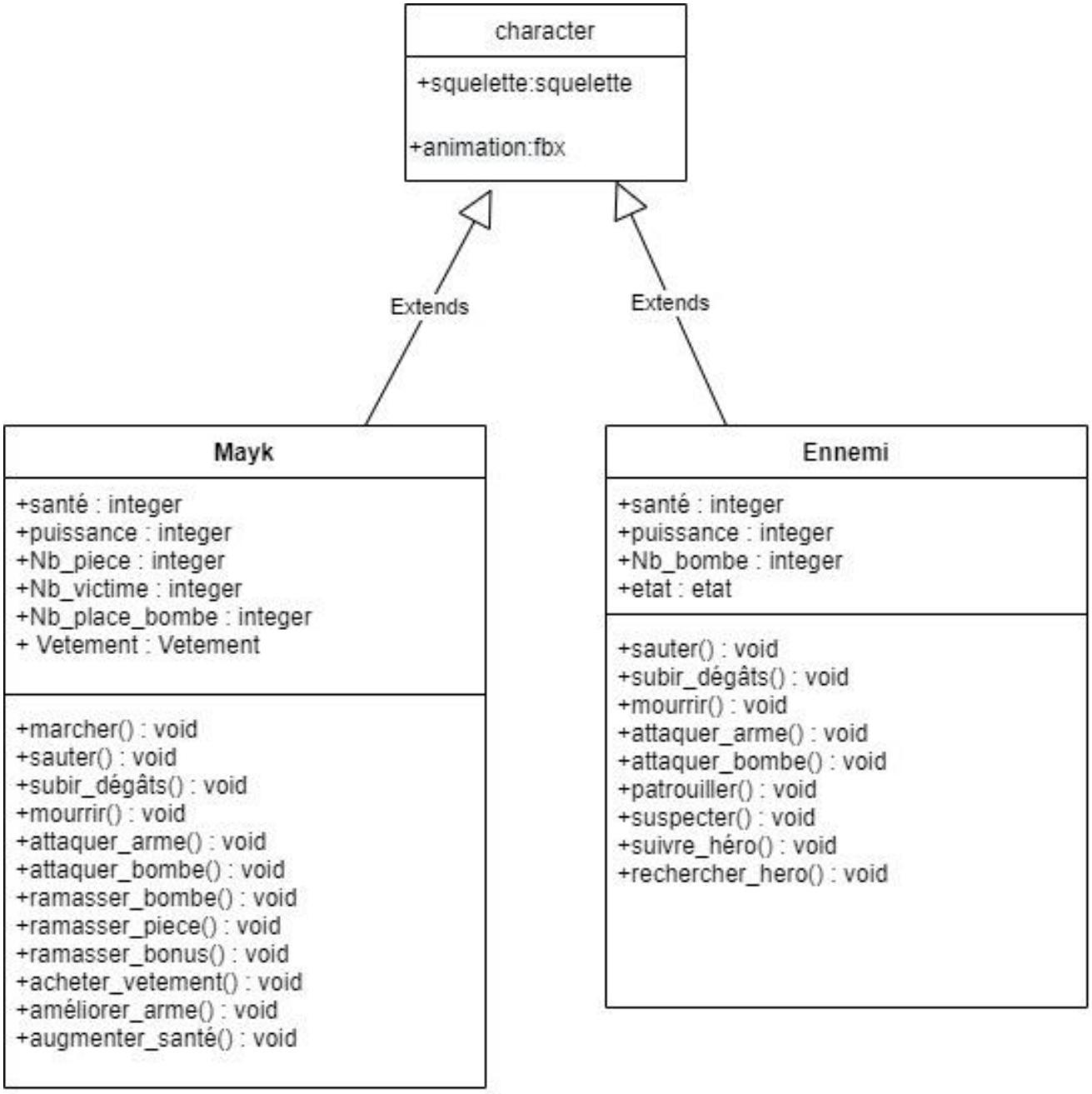


Figure III-21. Diagramme de classe des caractères dans le jeu EGGS ADVENTURE.

La figure suivante représente le diagramme de classe de la classe Mayk noté avec 2 dans la figure III-20 en faisant le lien avec la classe vêtement qui définit tous les objets qui peuvent être rattaché au caractère pour former les vêtements du personnage.

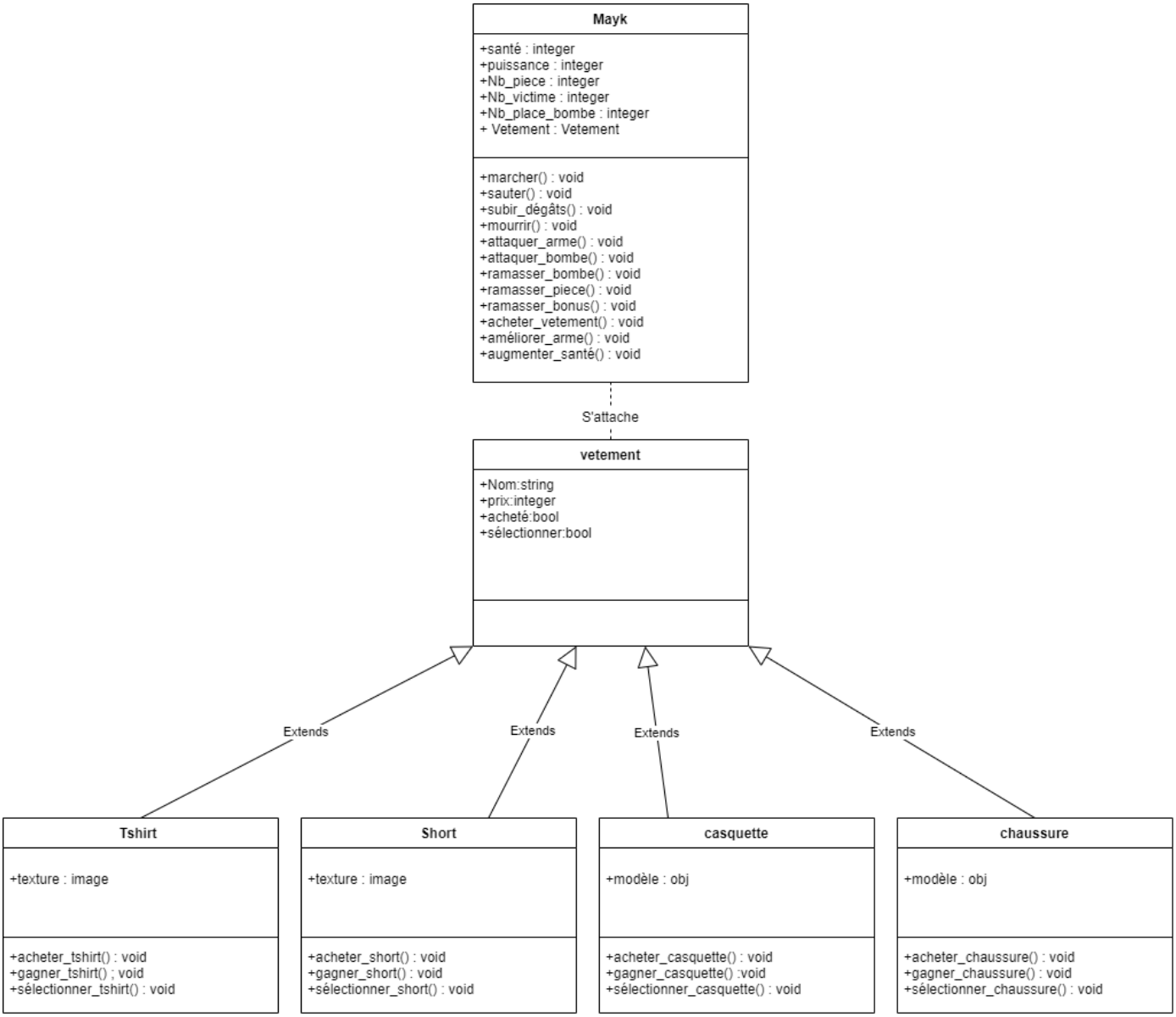


Figure III-22. Diagramme de classe de la classe Mayk dans le jeu EGGS ADVENTURE.

La figure suivante détaille les différentes classes notées avec 3 dans la figure III-20 qui sont l'arme principale, les œufs et les pièces.

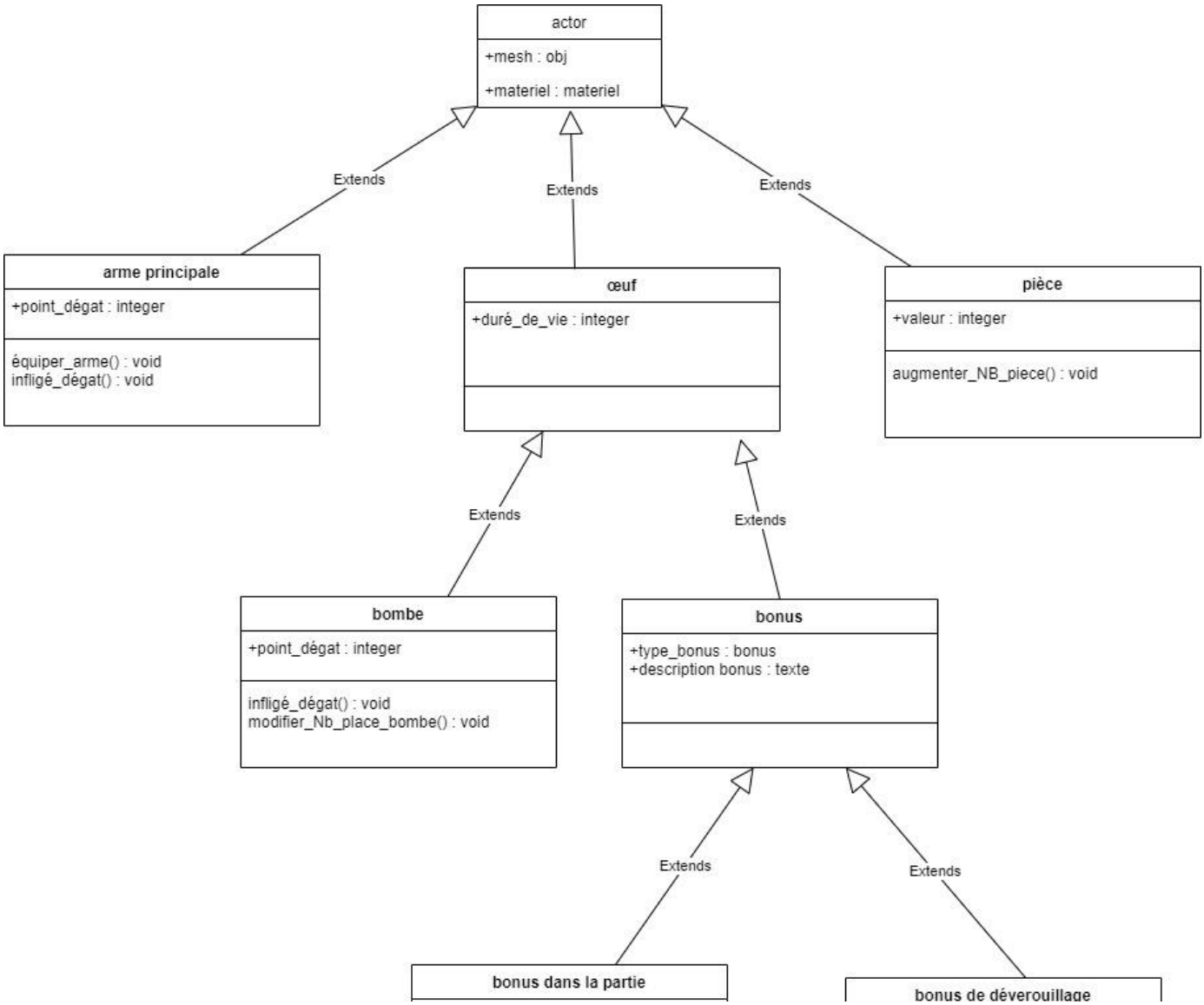
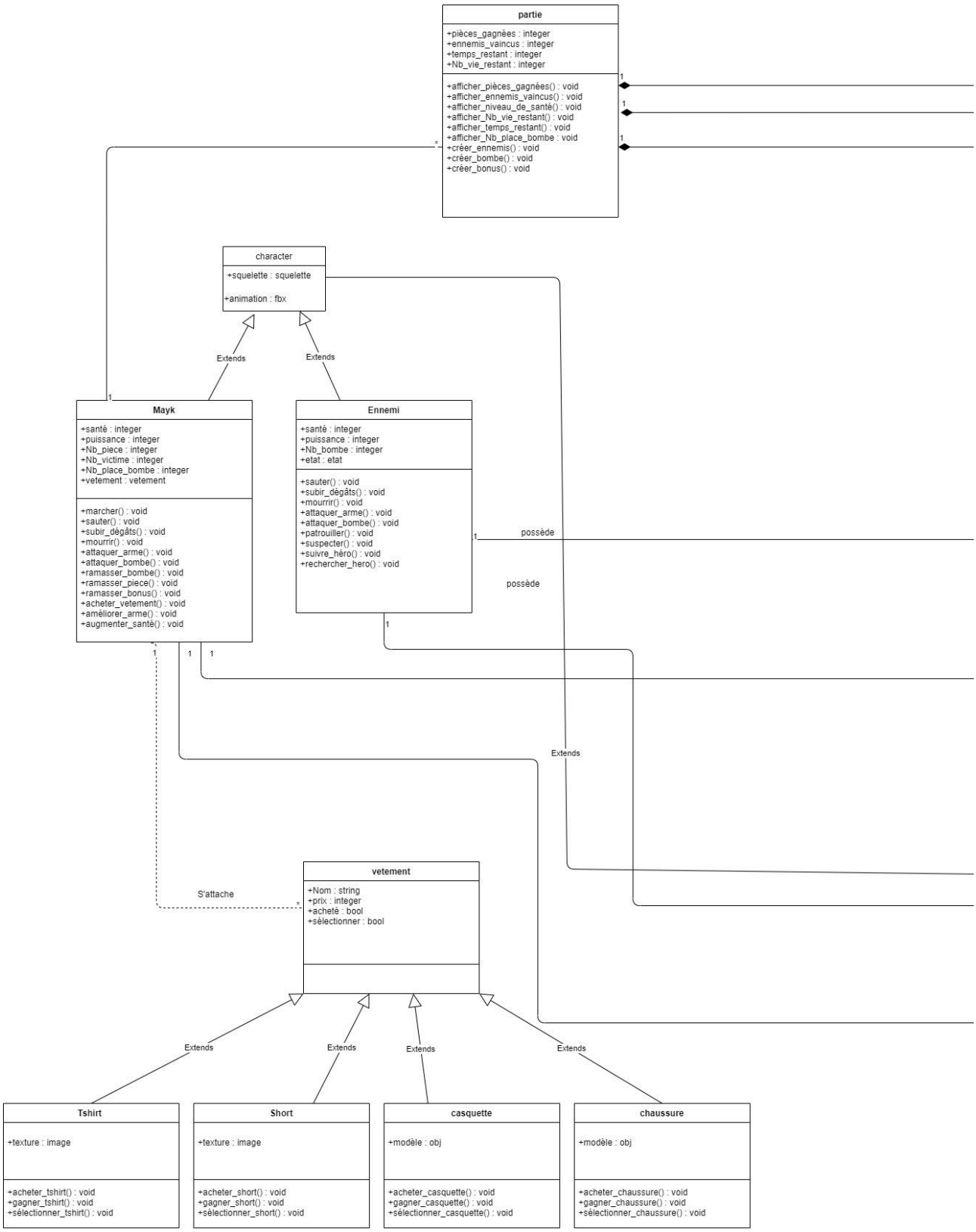


Figure III-23. Diagramme de classe des acteurs secondaires dans le jeu EGGS ADVENTURE.

La figure III-24 représente un diagramme global des classes les plus importante dans EGGS ADVENTURE.



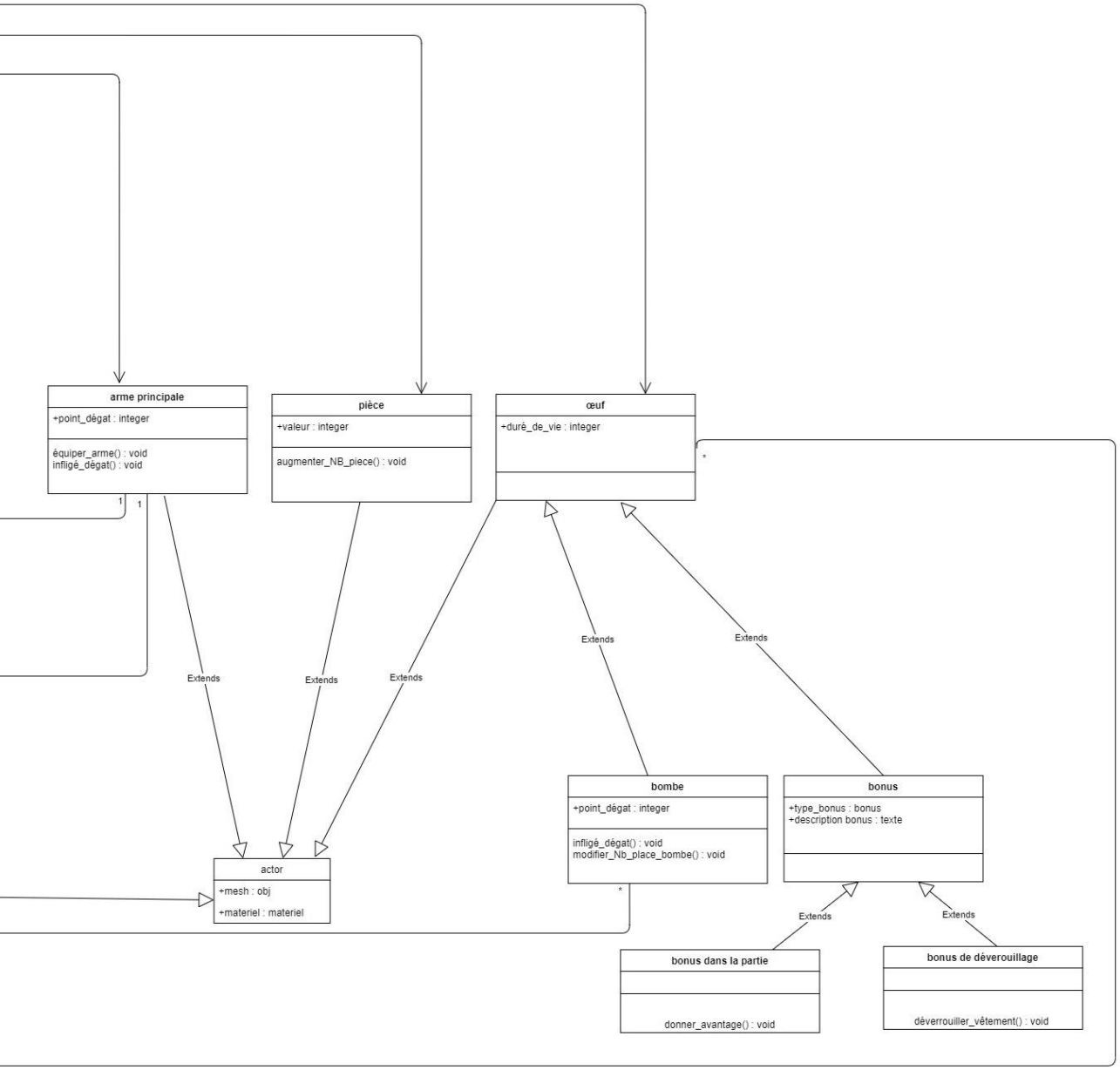


Figure III-24. Diagramme global des classes dans le jeu EGGS ADVENTURE.

La figure suivante représente le diagramme de cas d'utilisation de l'utilisateur.

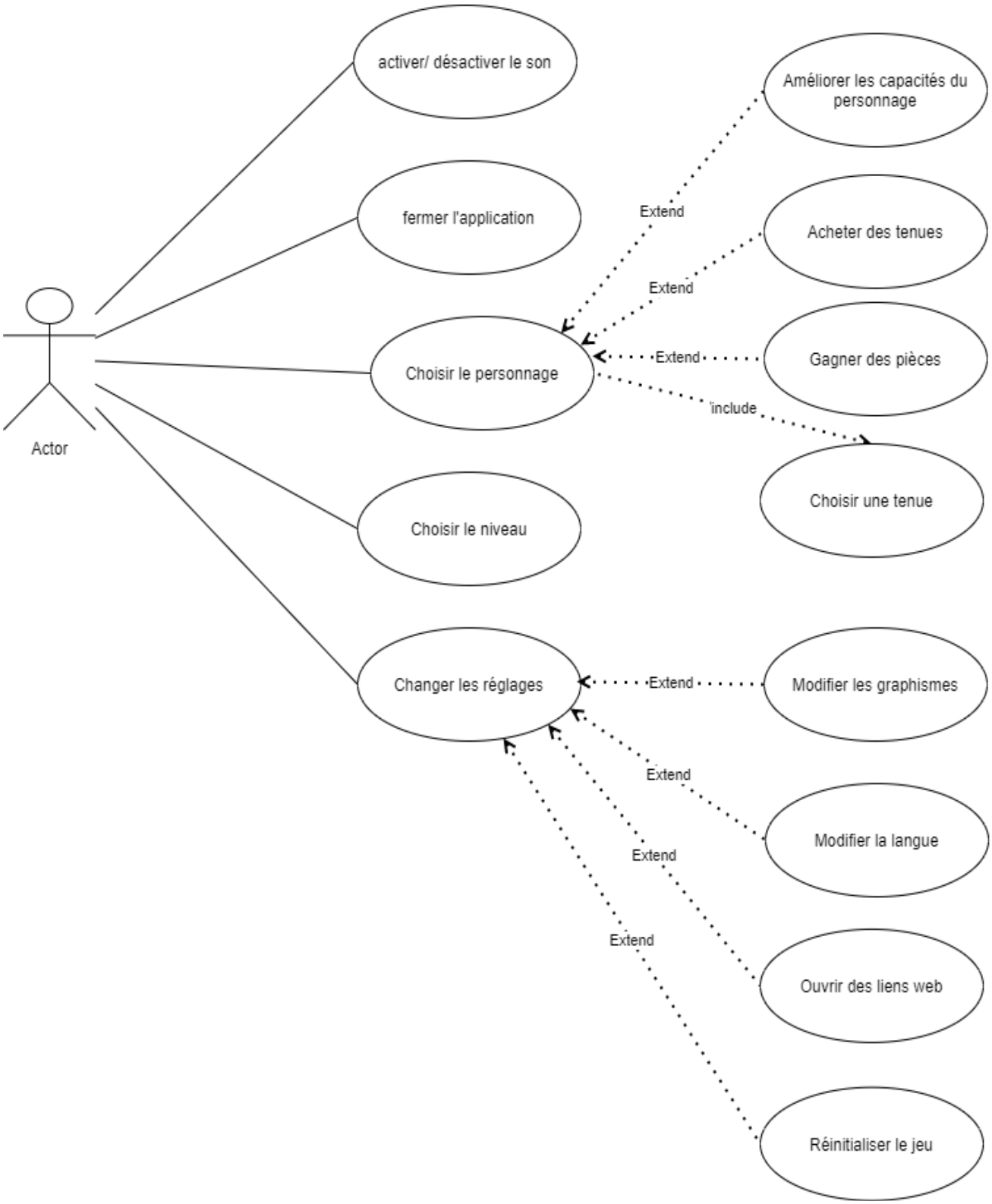


Figure III-25. Diagramme des cas d'utilisation de l'utilisateur.

Les figures III-26, III-27, III-28 représentent les différents diagrammes de séquence des différents menus dans l'interface utilisateur.

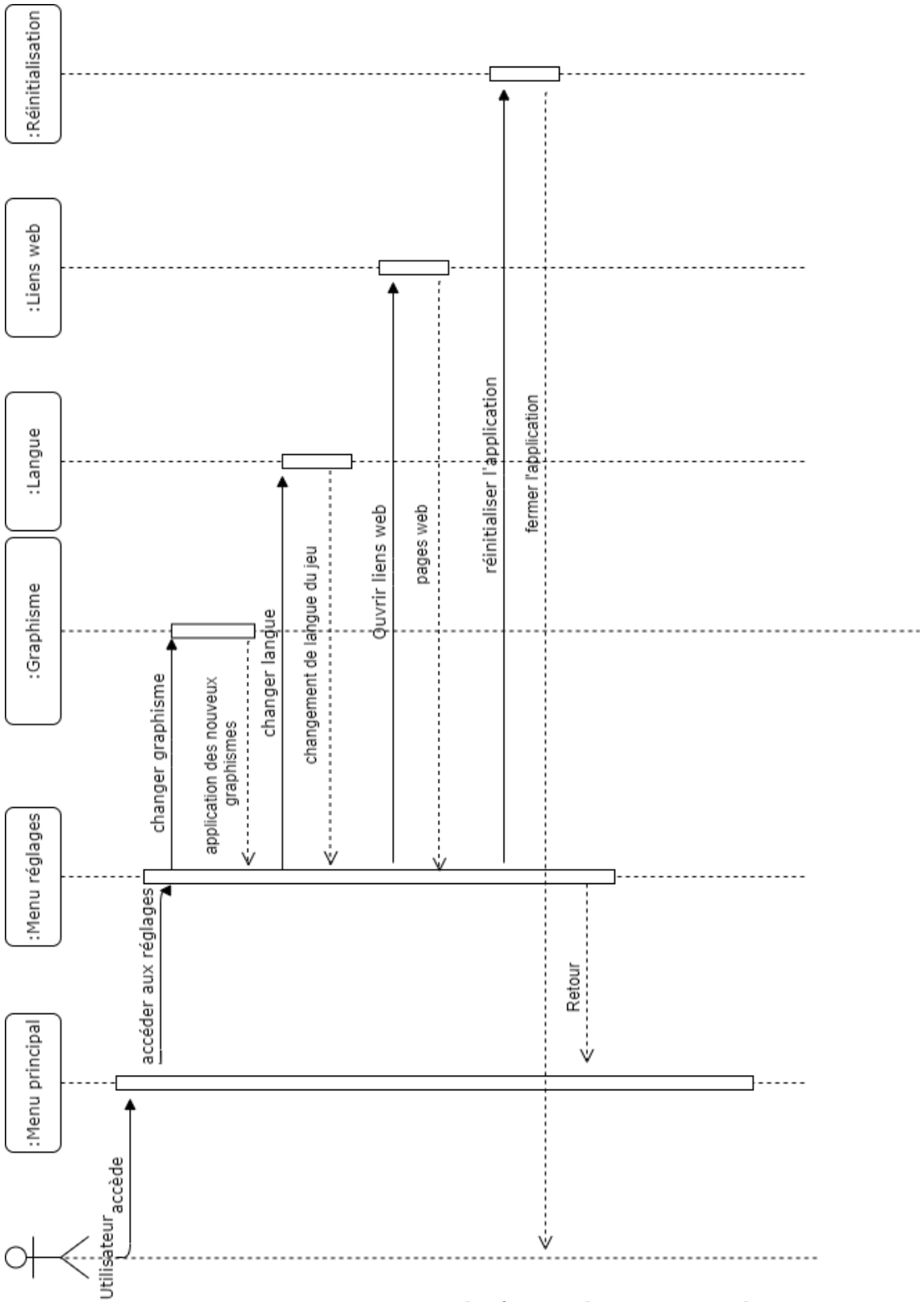


Figure III-26. Diagramme de séquence du menu principal.

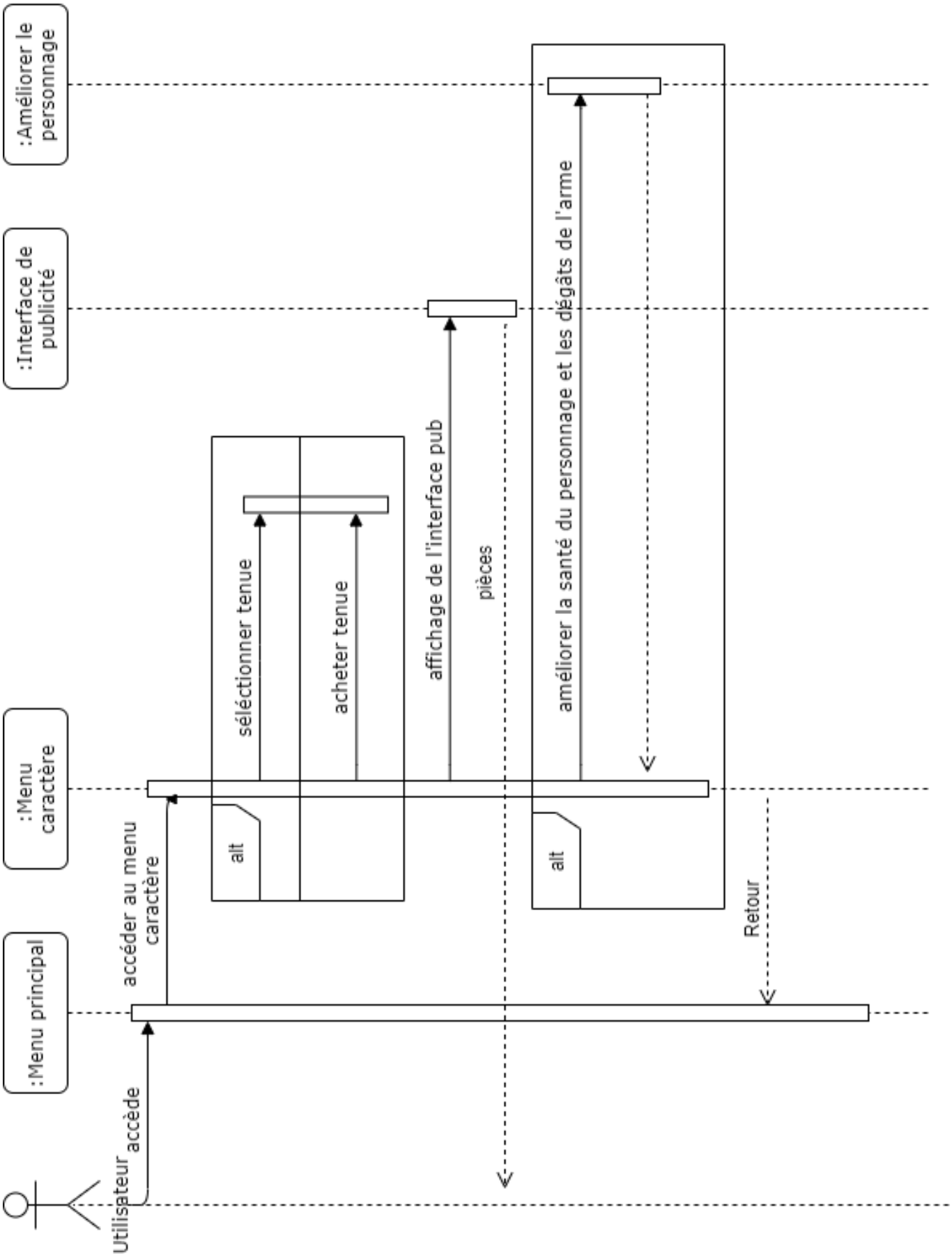


Figure III-27. Diagramme de séquence du menu caractère.

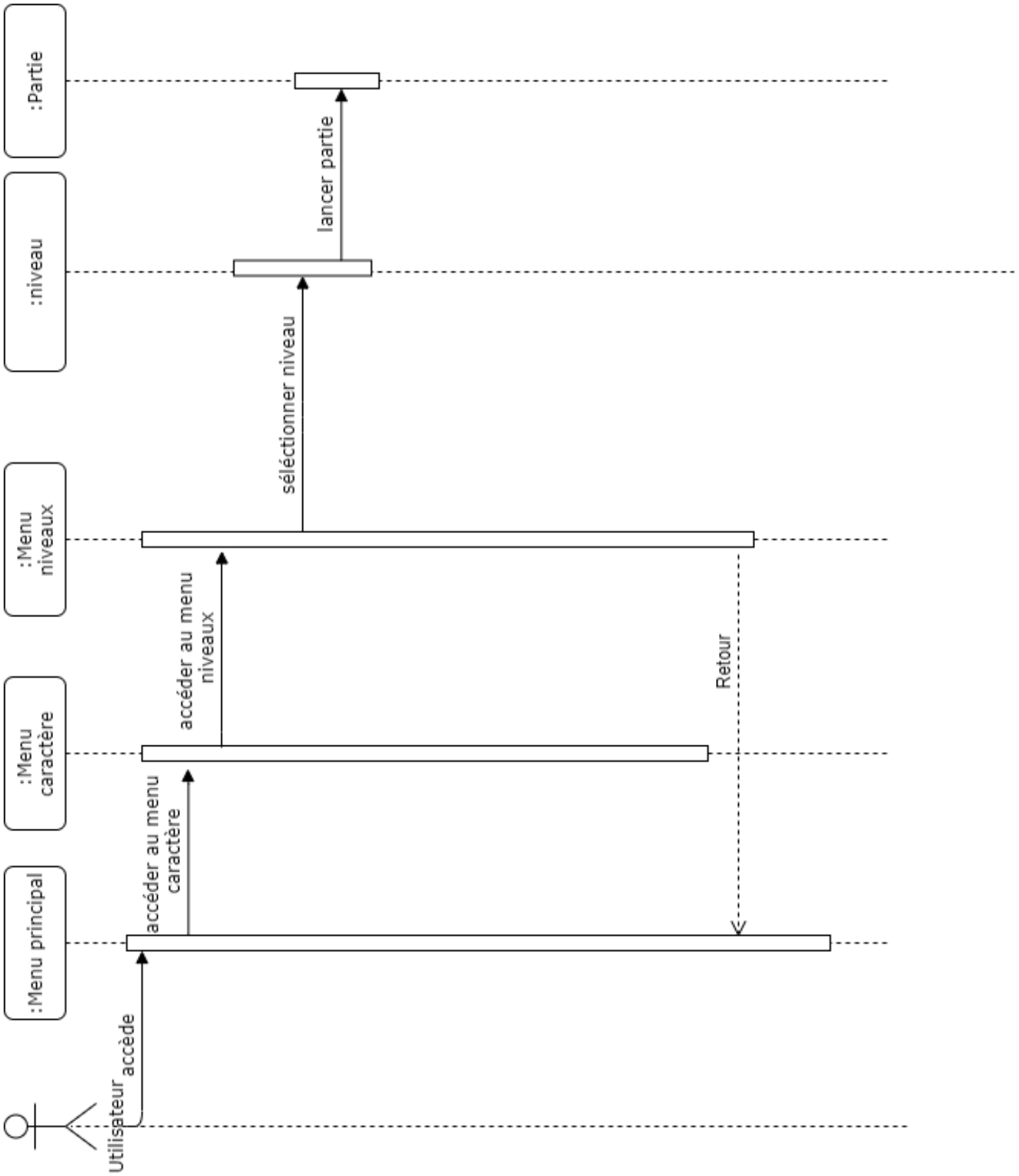


Figure III-28. Diagramme de séquence du menu niveau.

5. Conclusion

Dans ce chapitre, nous avons détaillé le document de conception du jeu EGGS ADVENTURE et ses différentes parties.
Ensuite nous avons proposé une hiérarchisation des différentes classes de notre jeu ainsi que des diagrammes de classe de quelques classes les plus importantes du jeu. Enfin on a proposé des diagrammes de cas d'utilisation et de séquence des différentes interfaces qui composent le menu utilisateur.
Dans le prochain chapitre et en se basant sur cette partie, nous allons décrire toutes les étapes qui ont été nécessaires à la réalisation du jeu EGGS ADVENTURE.

Chapitre IV : Réalisation du jeu EGGS ADVENTURE

1. Création et mise en place des éléments graphiques

1.1 Introduction

Dans cette partie du chapitre, nous allons présenter les différents logiciel et outils utilisés pour la création des éléments et objets présents dans le jeu EGGS ADVENTURE, et enfin nous détaillerons les étapes de création et de mise en place de ces éléments dans notre jeu.

1.2 Environnement et outils de développement

Autodesk 3ds Max

Autodesk 3ds Max [11] (ou simplement 3ds Max ou 3ds) est un logiciel de modélisation et d'animation 3D, développé par la société Autodesk.

3ds Max offre un large éventail de fonctionnalités : la modélisation polygonale, l'animation, la simulation, les effets spéciaux et le rendu réaliste.

Figure IV-1. Logo d'Autodesk 3ds Max

Adobe Photoshop et Illustrator

Adobe [12] Photoshop est un logiciel de retouche, de traitement et de dessin assisté par ordinateur. Édité par Adobe, il est principalement utilisé pour le traitement des photographies numériques, mais sert également à la création d'image.

Adobe Illustrator est un logiciel de création graphique vectorielle qui offre des outils de dessin vectoriel très complets

Adobe photoshop est Adobe Illustrator sont très utilisés dans le développement des jeux vidéo pour :

- La création des textures pour les objets 3D.
- La création des menus
- Générer les icones utilisées dans les menus.
- La création de calques et d'objets 2D.

Figure IV-2. Logo d'Adobe Photoshop et d'Adobe Illustrator.

Moteur Unreal Engine

Unreal Engine abrégé en UE est un moteur de jeu vidéo propriétaire développé par Epic Games [13] dans les débuts des années 90.

Unreal Engine a été développé par l'actuel directeur d'Epic Games Tim Sweeney avec quelques ingénieurs pour permettre aux développeurs de jeux au sein de la société de créer un jeu de tir à la première personne qui peut rivaliser avec le jeu Doom [14] édité par Id Software.

Le 19 mars 2014, Epic Games annonce qu'Unreal sera disponible pour le grand public au prix de 19 $ par mois, ainsi que 5 % de royalties.

Actuellement l'utilisation d'Unreal Engine est gratuite, mais avec une royaltie de 5% pour tous chiffre d'affaire dépassant le million de dollars.

Unreal est actuellement à sa quatrième version. Souvent mis à jour, la cinquième version du moteur est prévue pour l'année 2021.

Figure IV-3. Logo d'Unreal Engine.

Fonctionnalités et caractéristiques d'Unreal

Unreal mis à disposition de l'utilisateur une large gamme d'outils intégrés au moteur pour améliorer et accélérer le processus de développement des jeux.

a) Visuel scripting 'Blueprint'

L'une des caractéristiques les plus importantes d'Unreal c'est la possibilité de programmer les différentes classes et logiques du jeu grâce au langage de programmation C++, ainsi qu'avec un système de script visuel appelé *'Blueprint'*

Les blueprints permettent de faire appel à des fonctions, des macros et des parties du programmes grâce à des nœuds qui sont connectés entre eux par des connections sous forme de ligne.

Les blueprints peuvent remplacer la programmation traditionnelle en C++ dans presque tous les aspects de la création d'un jeu sur Unreal, mais la programmation en C++ est actuellement nécessaire pour implémenter des fonctionnalités plus avancées comme le multijoueur en ligne.

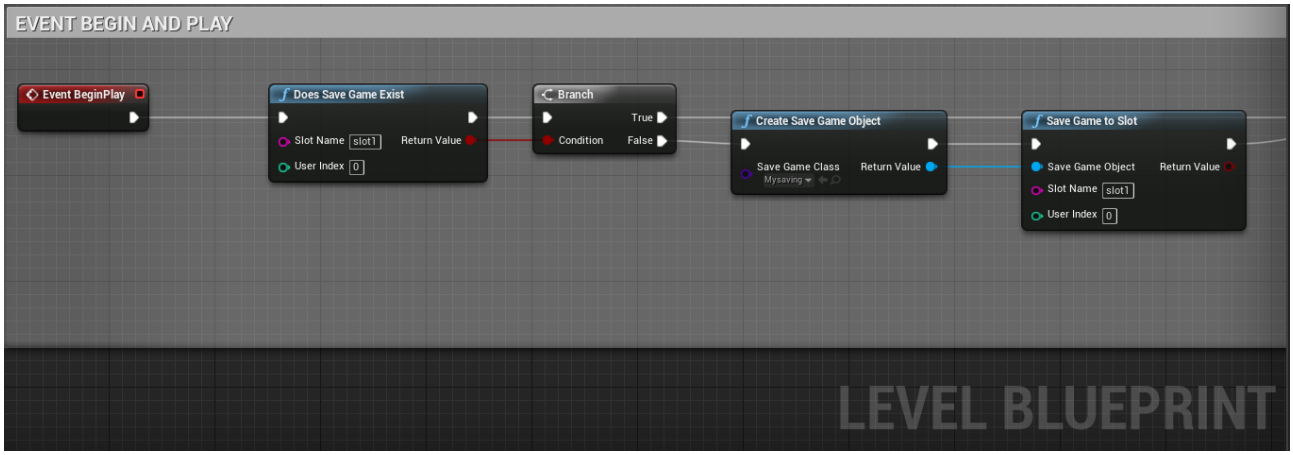


Figure IV-4. Exemple de blueprint.

b) L'intelligence artificielle.

« La logique vous mènera d'un point A à B, l'imagination vous mènera partout ». Les propos d'Einstein résument parfaitement l'univers de l'intelligence artificielle dans le monde des jeux.

Un caractère est dépourvu d'un sens logique, il ne peut être ni source de changement ni influenceur dans l'univers s'il n'est pas contrôlé sois par le joueur lui-même sois par une intelligence tierce qui prendra en charge tous les caractères non jouables par le joueur.

L'intelligence artificielle donne un comportement logique à un caractère basé sur une imagination préalable du développeur.

Le moteur Unreal utilise des **arbres de décisions** pour implémenter L'AI. À l'introduction du caractère AI dans la partie, le système d'intelligence Artificielle commence à exécuter le code contenu dans les différentes feuilles de l'arbre de décision partant de la gauche vers la droite grâce à des nœuds appelés composites.

L'arbre de décision utilise des variables propres à lui qui sont :

- Bool : booléen.
- Class : classe.
- Enum : énumération.
- Float : réel.
- Int : entier.
- Name : nom.
- Object : objet.
- Rotator : rotator.
- String : texte.
- Vector : vecteur.

Ces variables sont stockées dans une classe appelée ***BlackBoard*** qui est liée à l'arbre décision.

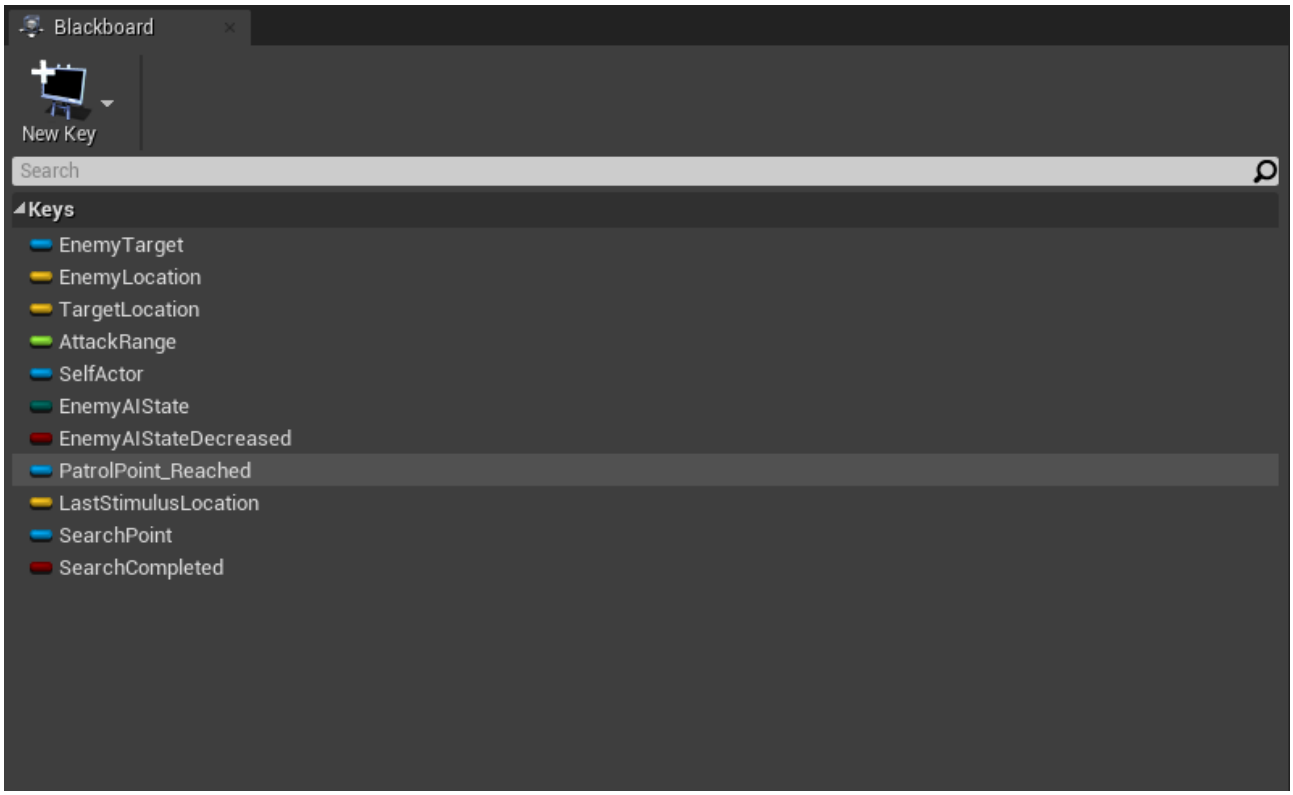


Figure IV-5. Exemple de blackBoard dans Unreal.

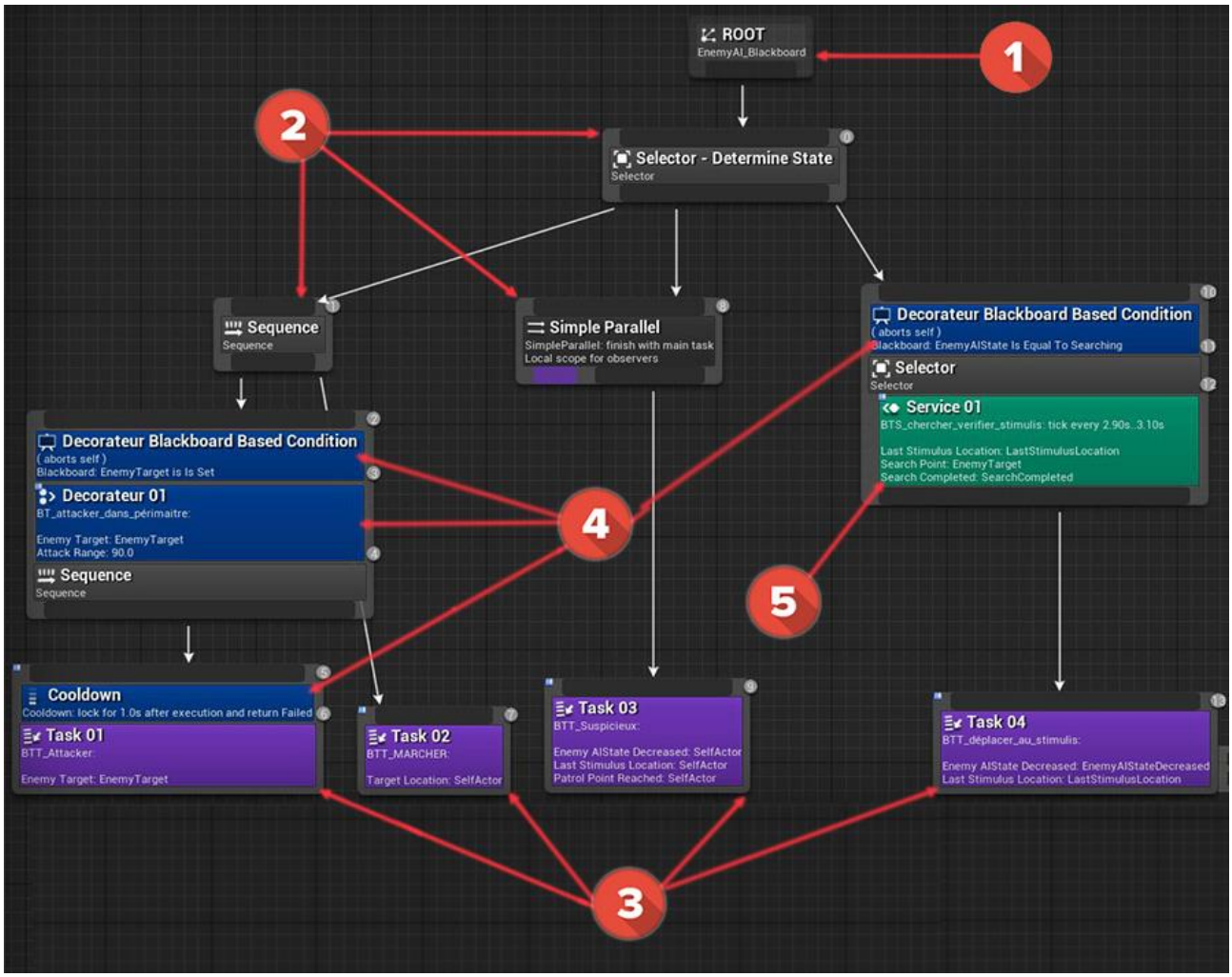


Figure IV-6. Exemple d'arbre de décision dans Unreal

1. Chaque arbre de décision est débuté par un nœud appelé ***ROOT*** ou racine indiqué par le numéro 1 dans la figure IV-6.

2. La Racine est suivie par des nœuds ***COMPOSITES*** ou compositions ; qui est soit un ***SELECTOR*** ou sélecteur, soit un nœud ***SEQUENCE*** ou éventuellement d'un nœud ***PARALLELE*** et sont indiqués par le numéro 2 dans la figure IV-6.

   - Les nœuds de composition permettent de parcourir l'arbre selon des logiques différentes, nous pouvons utiliser une infinité de nœuds de composition qui s'enchaîne de haut en bas, de gauche à droite.
   - Un SELECTOR exécute ses nœuds enfants de gauche à droite jusqu'à ce que l'un de ses descendants finis par un succès.
   - Une SEQUENCE exécute ses nœuds enfants de gauche à droite jusqu'à ce que l'un de ses descendants échoue.

3. Les ***TASKS*** ou tâches sont les blueprints les plus importantes de l'arbre de décisions, souvent les derniers éléments d'une branche, elles permettent de coder et de faire exécuter des actions aux caractères contrôlés par l'AI. Elles sont indiquées par le numéro 3 dans la figure IV-6.

4. Les ***DECORATORS*** ou décorateurs, sont des blueprints attachés à un nœud composite ou à un nœud de tâche et définissent si un nœud ou la totalité d'une branche de l'arborescence peut s'exécuter. Ils sont indiqués par le numéro 4 dans la figure IV-6.

   Les plus utilisés sont :

   - Les décorateurs basés sur la vérification des valeurs des blackboard qui retourne un booléen.
   - Le décorateur *Cooldown* qui retourne un booléen faux à l'écoulement d'une durée précise après l'exécution de la tâche.
   - Le décorateur *Time Limit* qui retourne un booléen faux après une durée de temps depuis l'atteinte du nœud.
   - Les décorateurs issus d'un code personnalisé du développeur.

5. Les ***SERVICES*** ou services sont des blueprints souvent utilisés avec des composites ou des tâches et s'exécutent à des fréquences définies tant que leurs branches sont en cours d'exécution. Ils sont utilisés pour effectuer des vérifications et pour mettre à jour les valeurs du BlackBoard. Ils sont indiqués par le numéro 5 dans la figure IV-6.

1.3 Création des éléments graphiques du jeu

1.3.1. Scène d'introduction

La scène d'introduction est mise en place dans une scène sous Unreal, et se base essentiellement des concepts art de l'histoire du jeu.

Les poses du caractère ainsi que les différents éléments 3D qui sont spécifique à cette scène ont été réalisés avec le logiciel 3DS MAX.

Figure IV-7. Rendus finaux des concepts art 2,4,6 de l'introduction

1.3.2. Les vêtements de Mayk

La personnalisation du T-shirt et short du personnage Mayk se fait en appliquant les différentes textures conçues avec le logiciel Photoshop dans une zone qu'on appelle matériel sur la partie du caractère qui représente le T-shirt et le short.

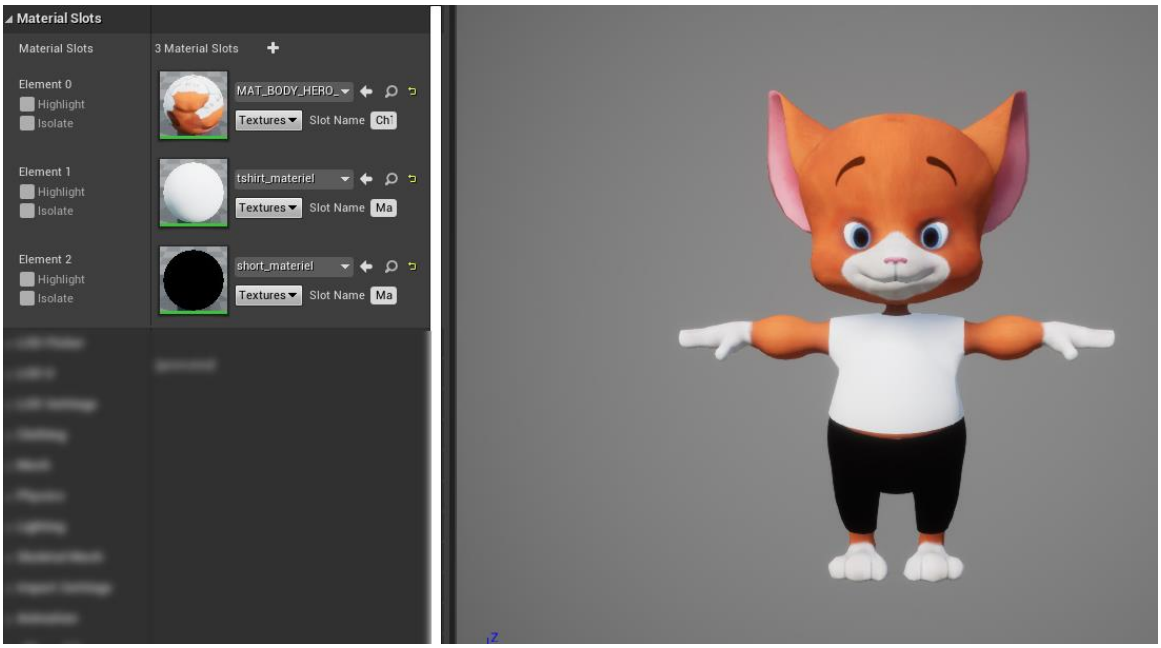


Figure IV-8. Menu d'assignation des différents matériels.

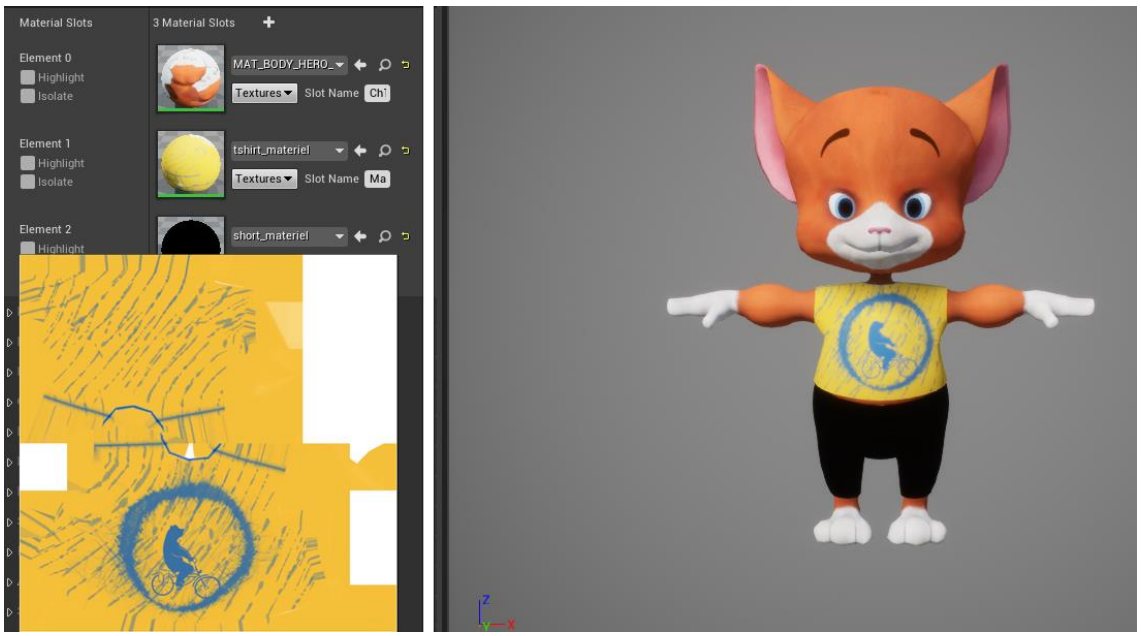


Figure IV-9. Exemple d'application d'une texture sur le personnage.

Les chaussures du personnage et des différents chapeaux sont des modèles 3D modélisés avec des logiciels 3D et texturés avec des textures issues du logiciel Photoshop. Ils sont attachés au personnage grâce à des fixations appelées '**Socket'.**

Plusieurs modèles sont dupliqués et existent avec des textures différentes appliquées avec la même méthode décrite précédemment.

Figure IV-10. Exemple d'attachement de chaussures et d'un chapeau au caractère.

1.3.3. Les œufs et pièces

Les œufs sont obtenus d'une déformation d'une sphère sous 3DS Max et l'application de différentes textures sur le modèle sous Unreal.

La couleur rose est dédiée aux bombes, la couleur rouge aux bonus dans le jeu et enfin la couleur dorée aux bonus débloquant des personnalisations.

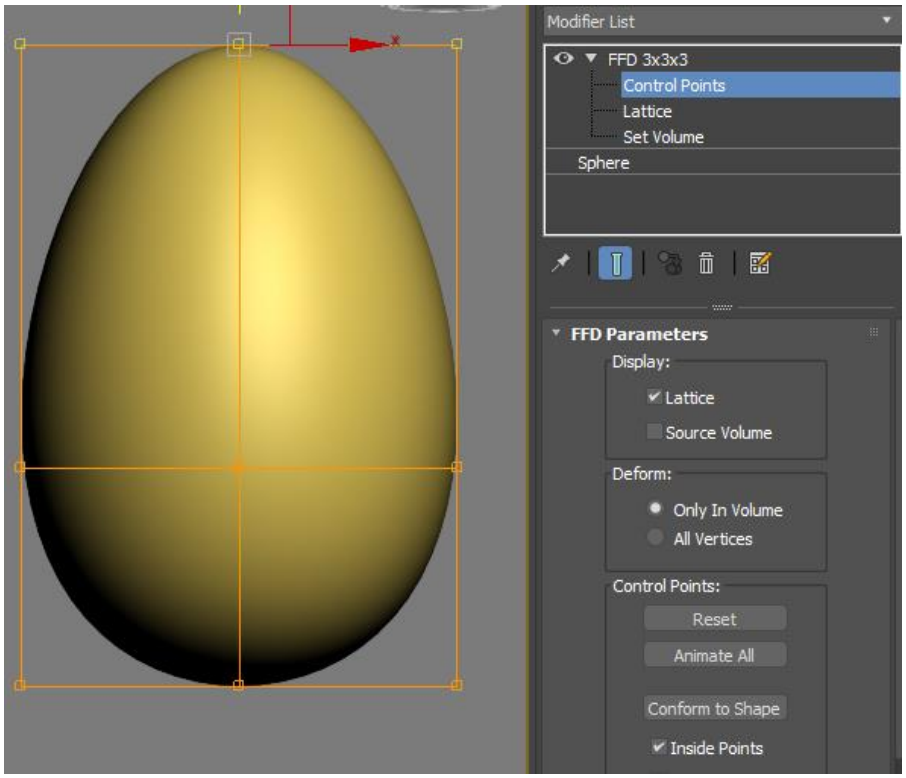


Figure IV-11. Modélisation d'un œuf sous 3DS Max.

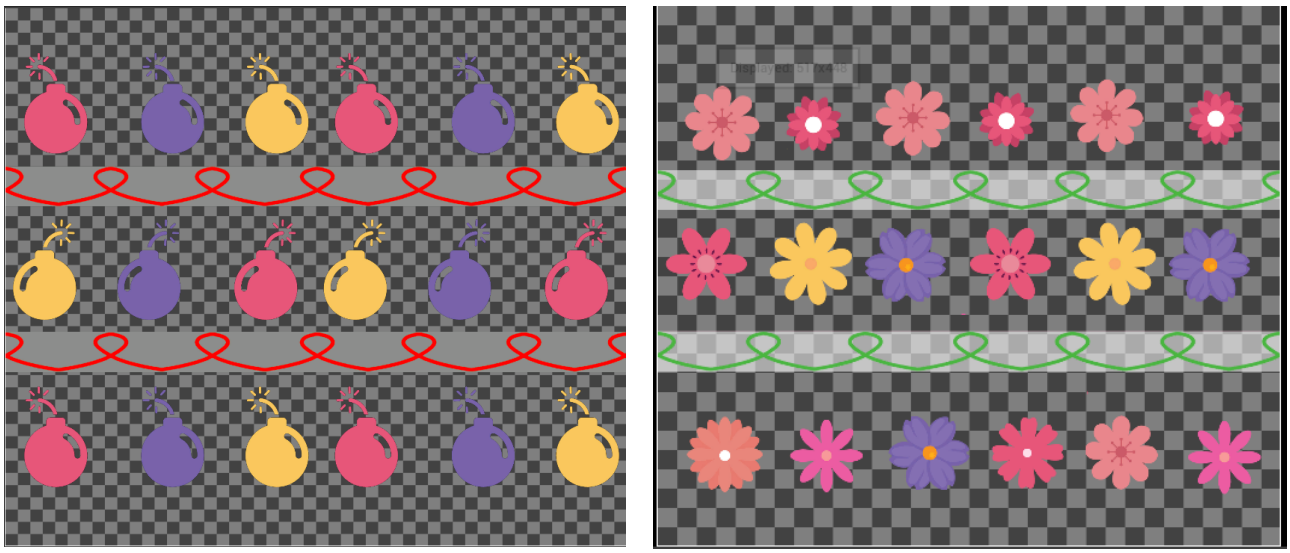


Figure IV-12. Différentes textures appliquées aux œufs dans Unreal.

Figure IV-13. Rendus réalistes des différents œufs présentent dans EGGS ADVENTURE.

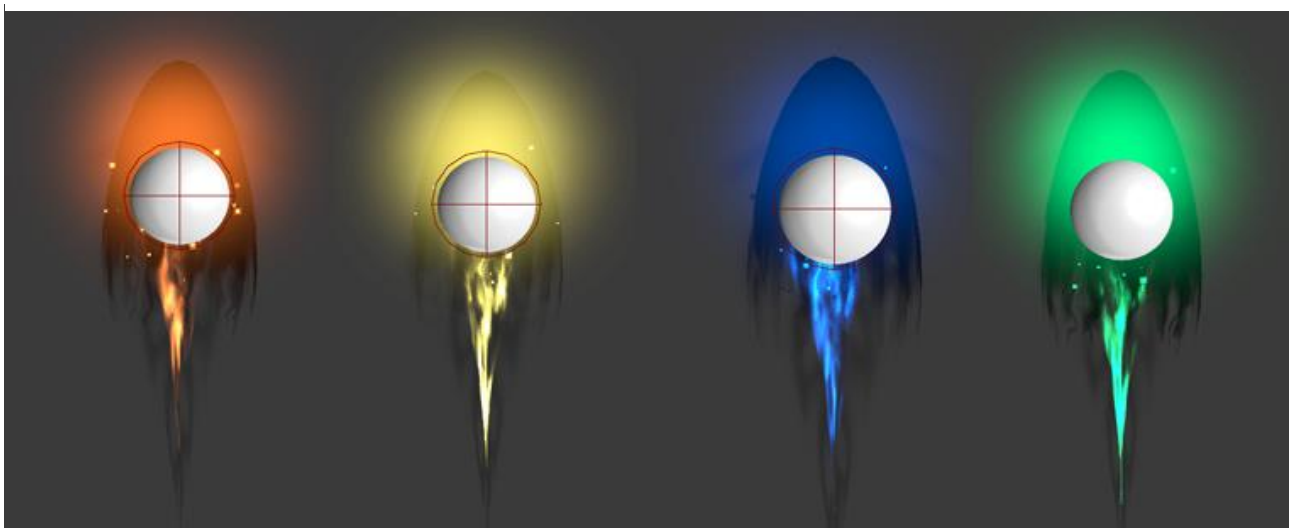


Figure IV-14. Rendu des différentes bombes présentent dans le jeu.

Les pièces sont obtenues grâce à une modélisation par extrusion d'un cercle pour la partie dorée et d'une modélisation CSG pour les trous en bleu.

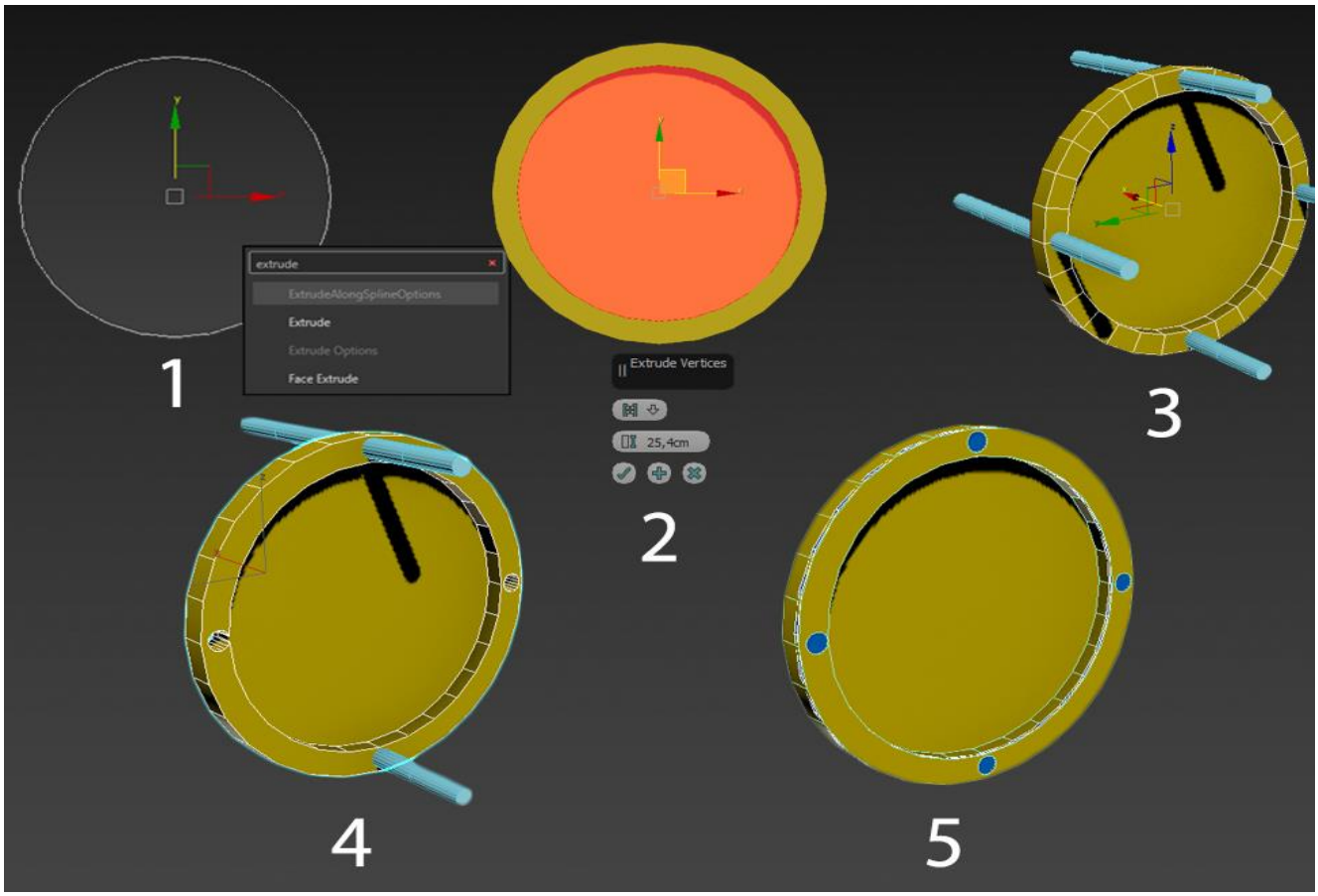


Figure IV-15. Etape de création de la pièce.

1.3.4. Les armes

Les armes sont modélisées grâce à la combinaison de plusieurs types de modélisation comme la modélisation CSG et par extrusion.

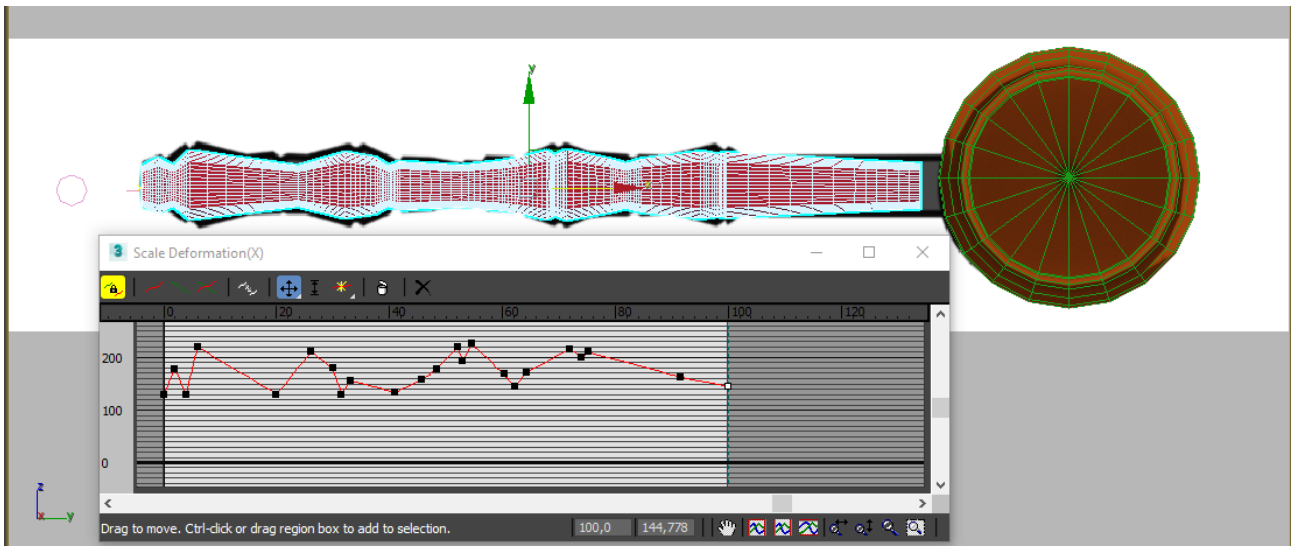


Figure IV-16. Modélisation de l'arme principale du héros dans 3DS Max.

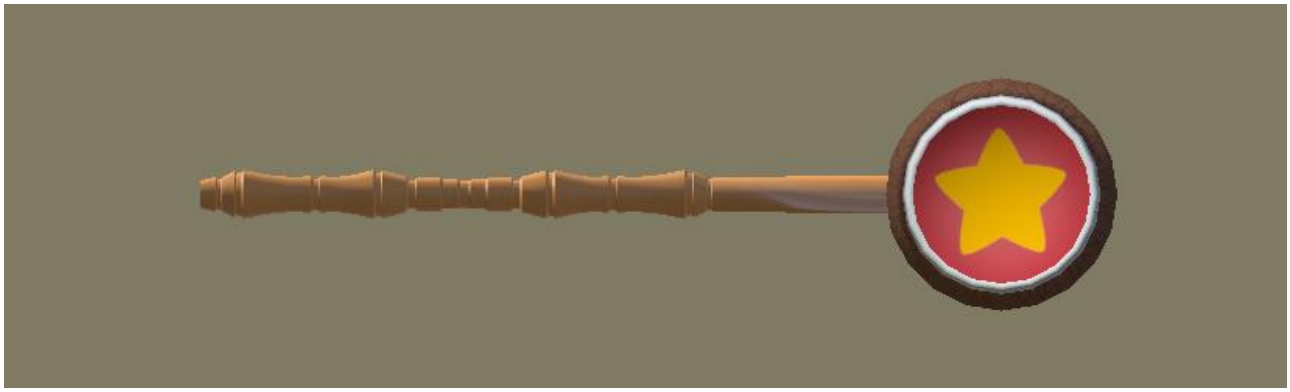


Figure IV-17. Rendu réaliste de l'arme principale du héros.

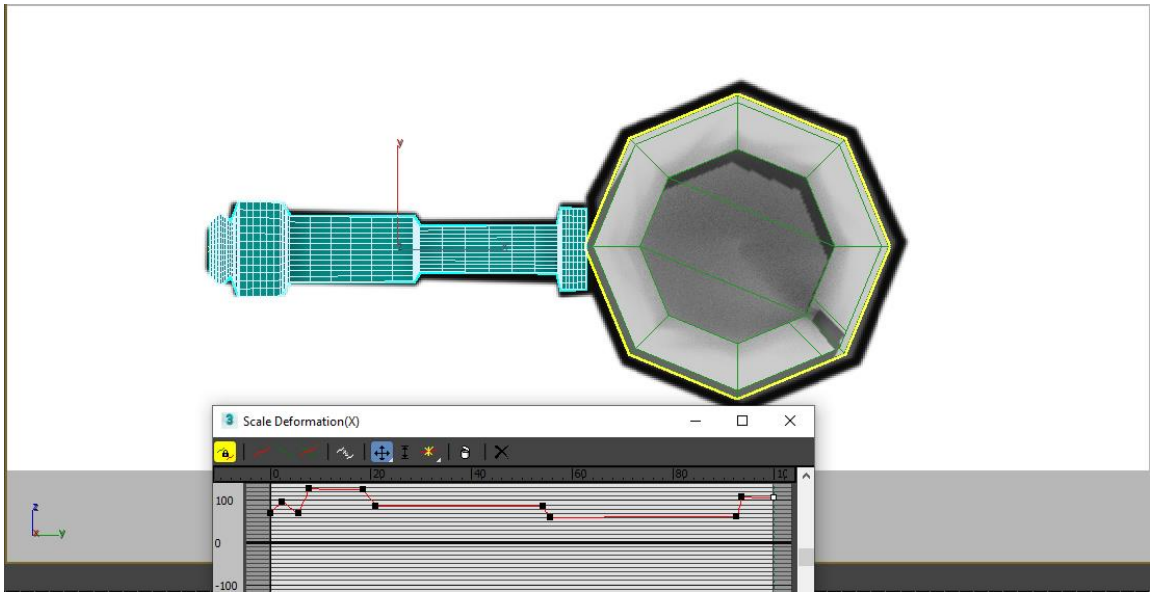


Figure IV-18. Modélisation de l'arme principale des ennemis dans 3DS Max

Figure IV-19. Rendu réaliste de l'arme principale des ennemis.

1.3.5. Fleurs et végétation

Afin d'optimiser le jeu pour une utilisation plus fluide sur des terminaux mobiles, les végétations sont conçues avec un simple rectangle à qu'est appliqué une texture de fleur ou d'herbe associée à une texture d'alpha pour découper les bords de la textures non voulus.

Dans la partie du matériel qui utilise cette texture, une logique de rotation du rectangle en direction de la caméra est utilisée afin que tous les rectangles soient toujours face à la caméra, ce qu'évite d'avoir des rectangles noirs et des végétations plates.

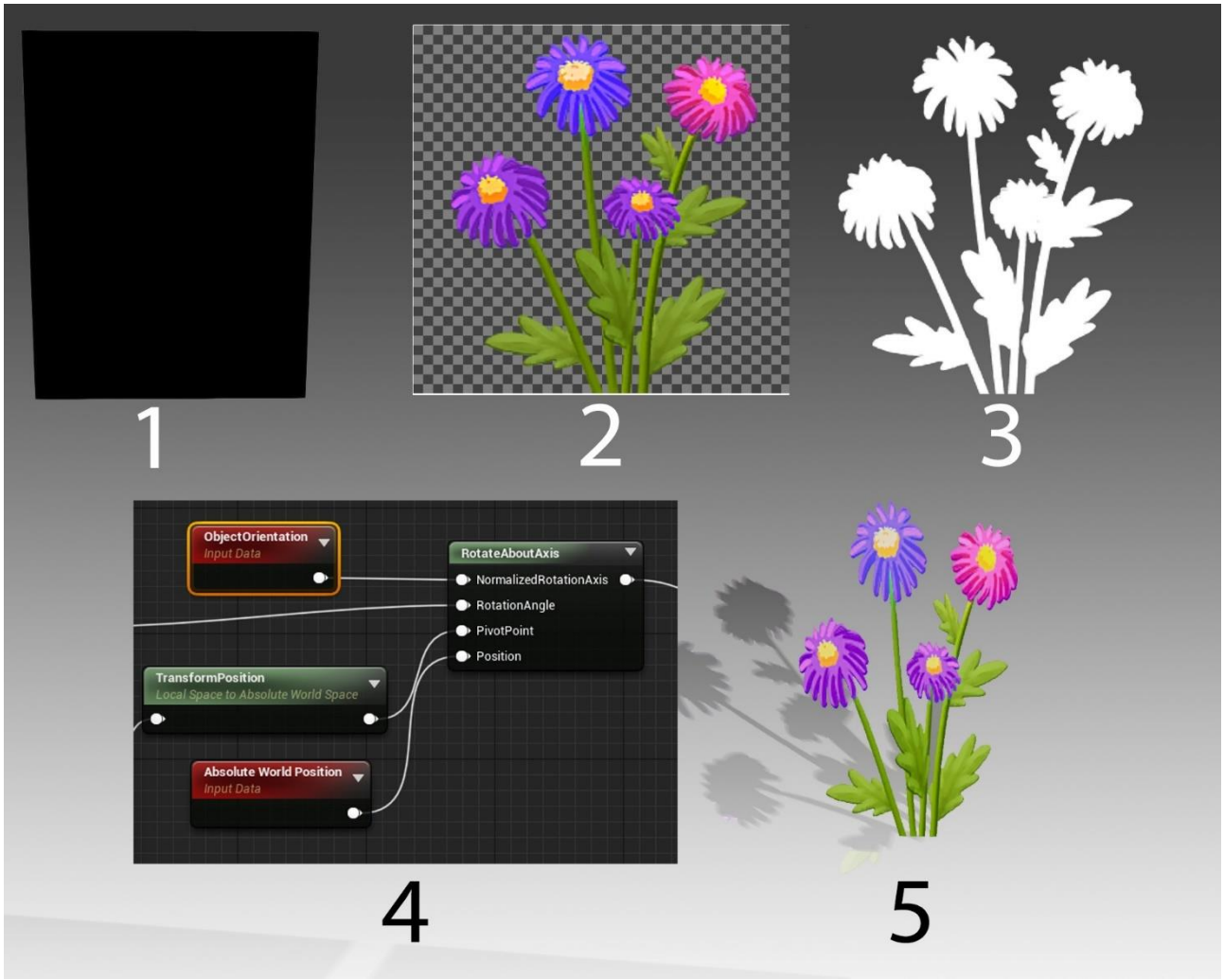


Figure IV-20. Etape de création d'un élément de végétation.

2. Animation des personnages :

Les personnages sont composés d'un objet 3D appelé *Skeletal Mesh* ou squelette mesh qui est associé à un système d'os appelé *skeleton*.

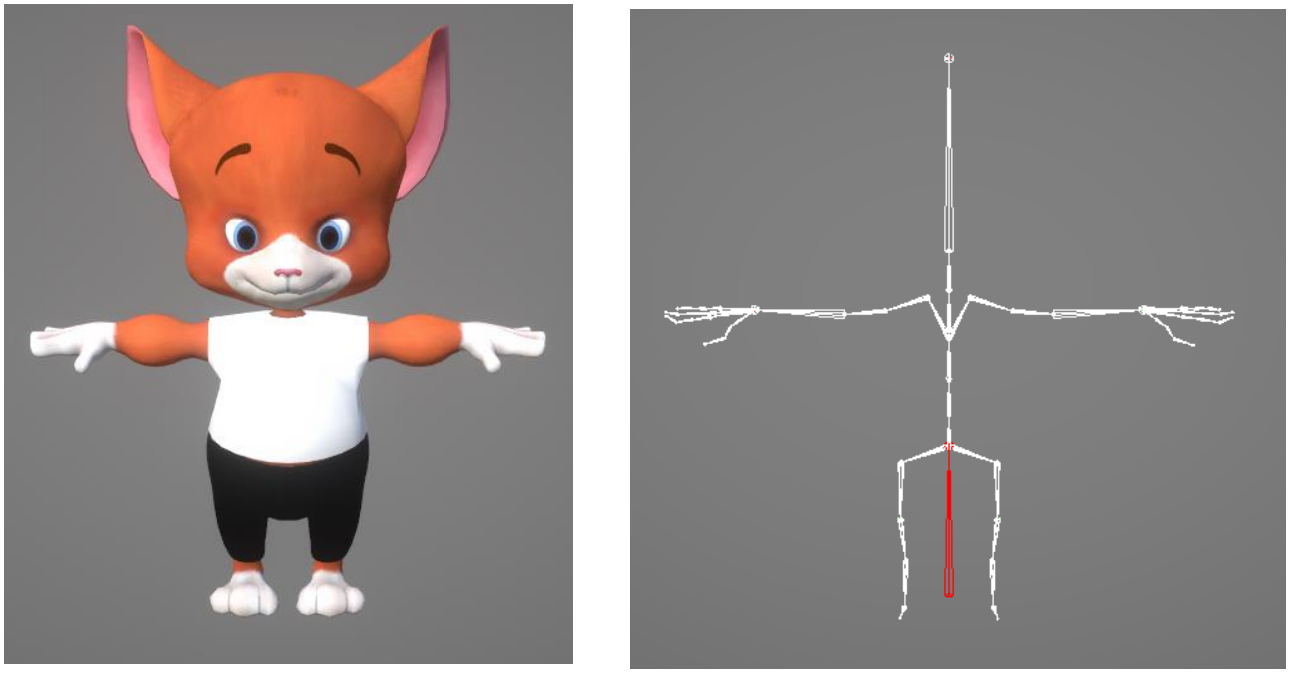


Figure IV-21. *Tpose* et *Bones* du personnage.

Le personnage peut se déplacer dans l'univers et exécuter des tâches propres à lui suivant des scripts propres à chaque personnage qui seront détaillé en bas, mais sans animations le personnage se déplacera avec la même pose par défaut qu'on voit en dessus.

Pour être le plus réaliste possible, des animations sont rajoutées aux personnages et qui seront jouées de façon continues pendant la période de jeu.

Chaque personnage est rattaché à un Blueprint qu'on appelle *Animation Blueprint* qui gère toutes les animations du personnage.

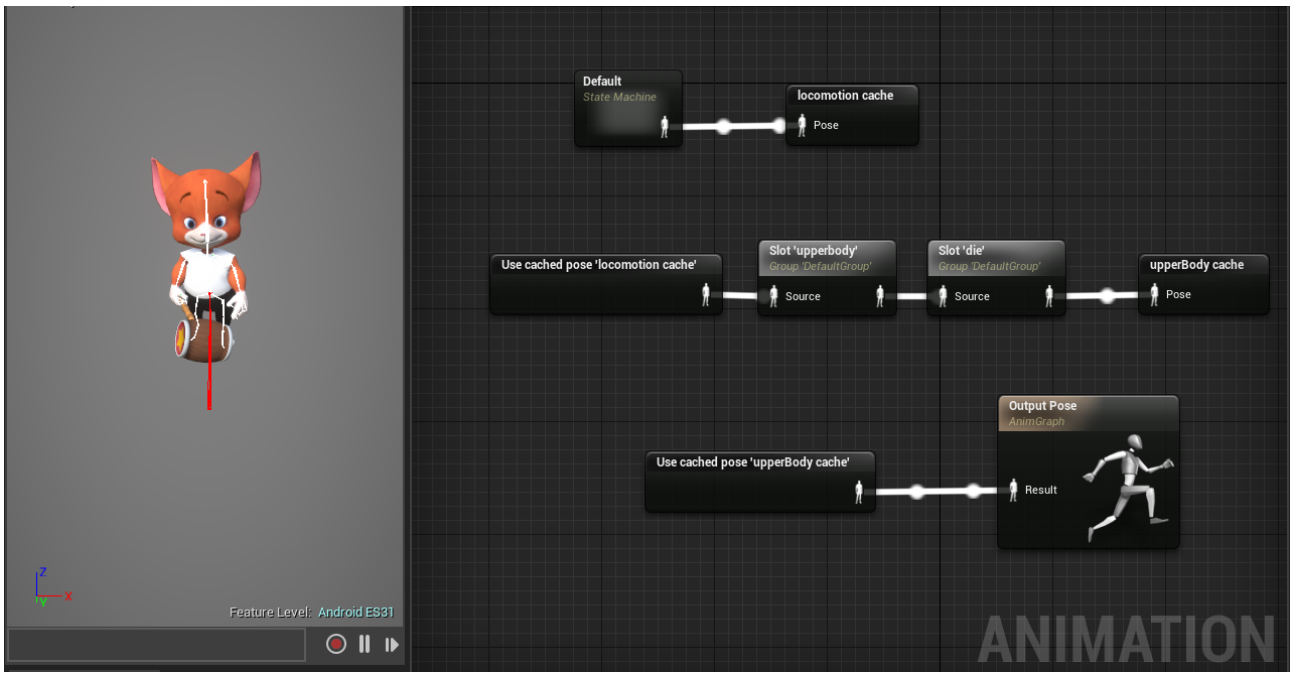


Figure IV-22. Blueprint Animation dans Unreal.

L'animation Blueprint contient par défaut Un nœud *State Machine* où nous définissons les animations standard que le joueur peut effectuer et les conditions pour lesquelles il peut passer d'une animation à une autre.

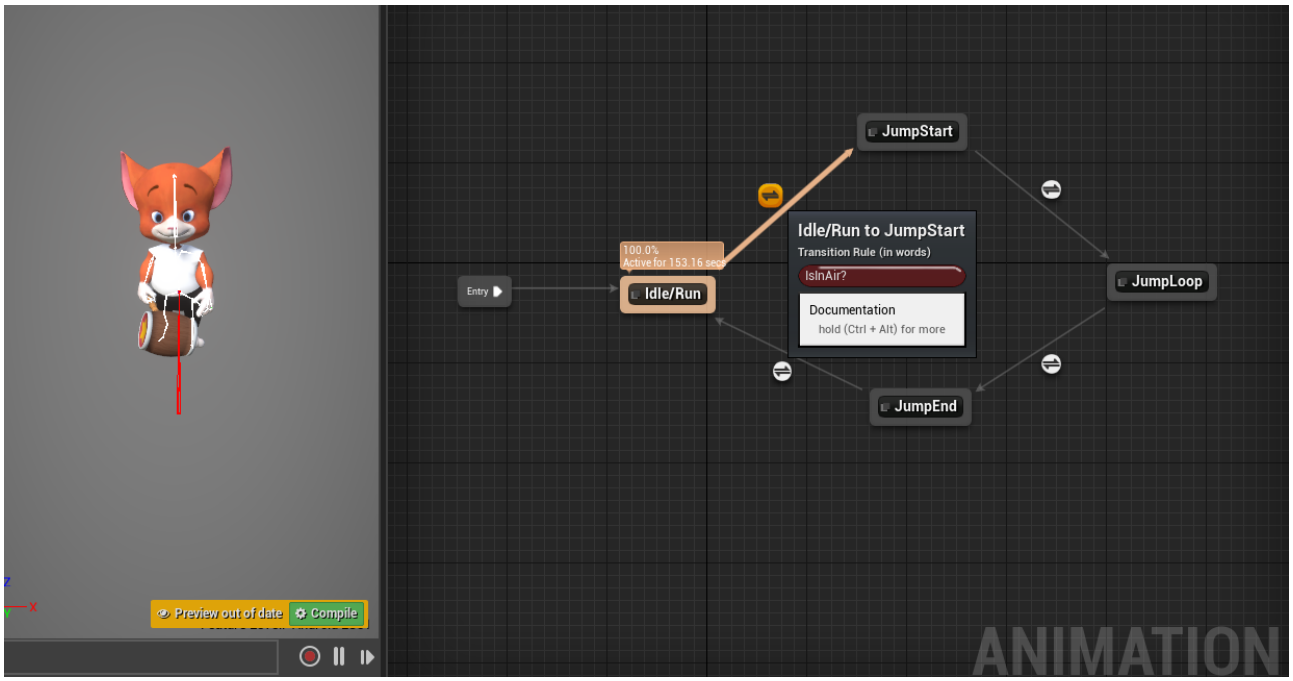


Figure IV-23. State machine dans un animation Blueprint.

Figure IV-24. Les animations principales dans le jeu.

3. Script des classes :

Tous les personnages et les objets dynamique du jeu possèdent une partie code par laquelle le joueur pourra interagir avec ce dernier, cette partie du code est appelée soit par le joueur en demandant ou en faisant une action, soit par le moteur lui-même indépendamment du joueur en réponse à un changement dans la partie ou appelée dans période donnée.

1. Le héros

Le personnage principal du jeu contient une grande partie du script du jeu (sous forme de fonctions parfois organisées en blocs), qui permet au personnage de se déplacer dans l'environnement, d'exécuter des actions, et d'en subir d'autres :

- Les fonctions dans le bloc **Mouvement Input** (figure IV-25) permettent de déplacer le caractère dans l'environnement, tandis que la fonction **Jump** (figure IV-25) permet de faire sauter le caractère.
- La fonction **Event Tick** (figure IV-26) est appelée à chaque image permet au joueur d'orienter la caméra en utilisant le mouvement de la main sur l'écran tactile.

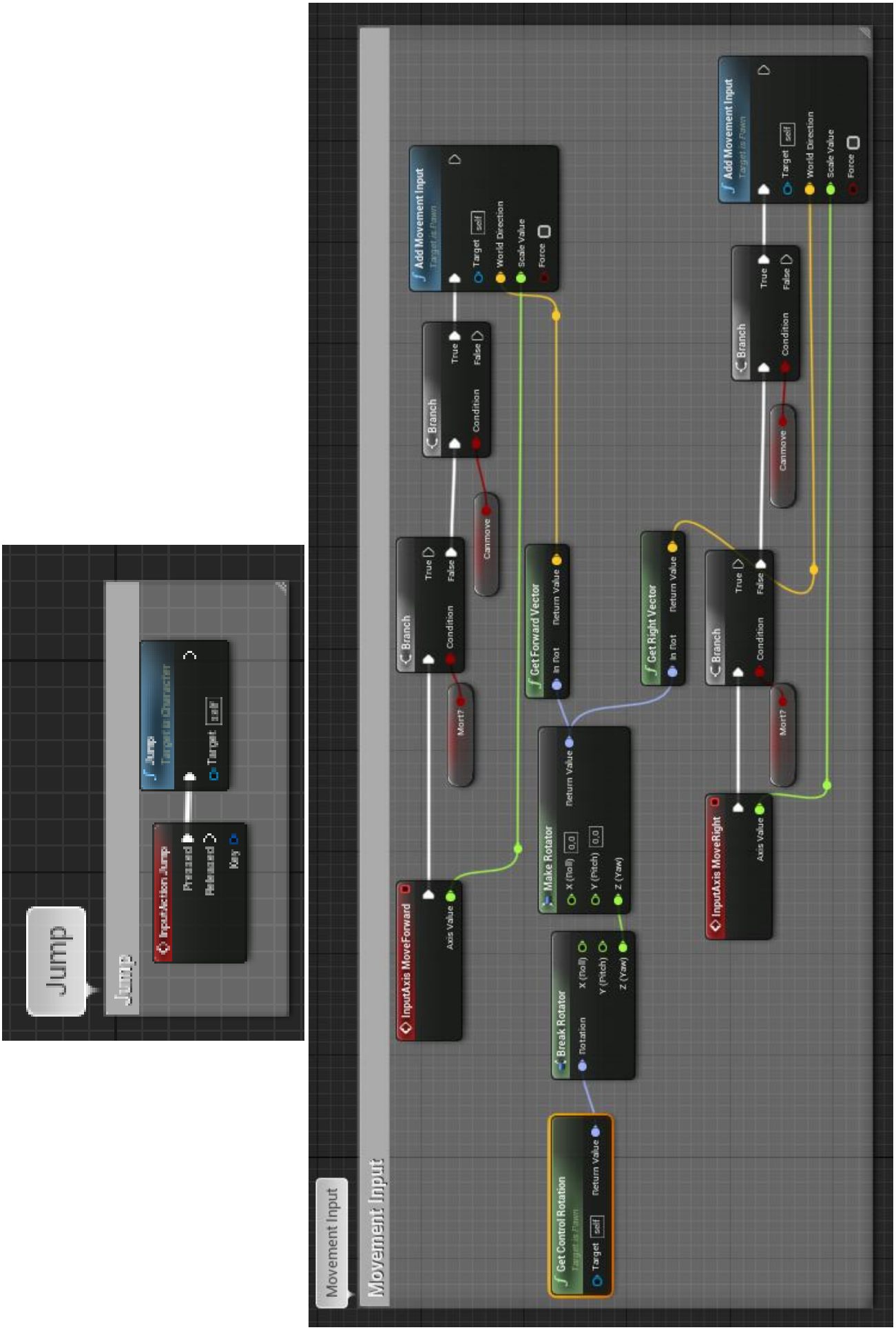


Figure IV-25. Blueprints sauter et déplacement du joueur.

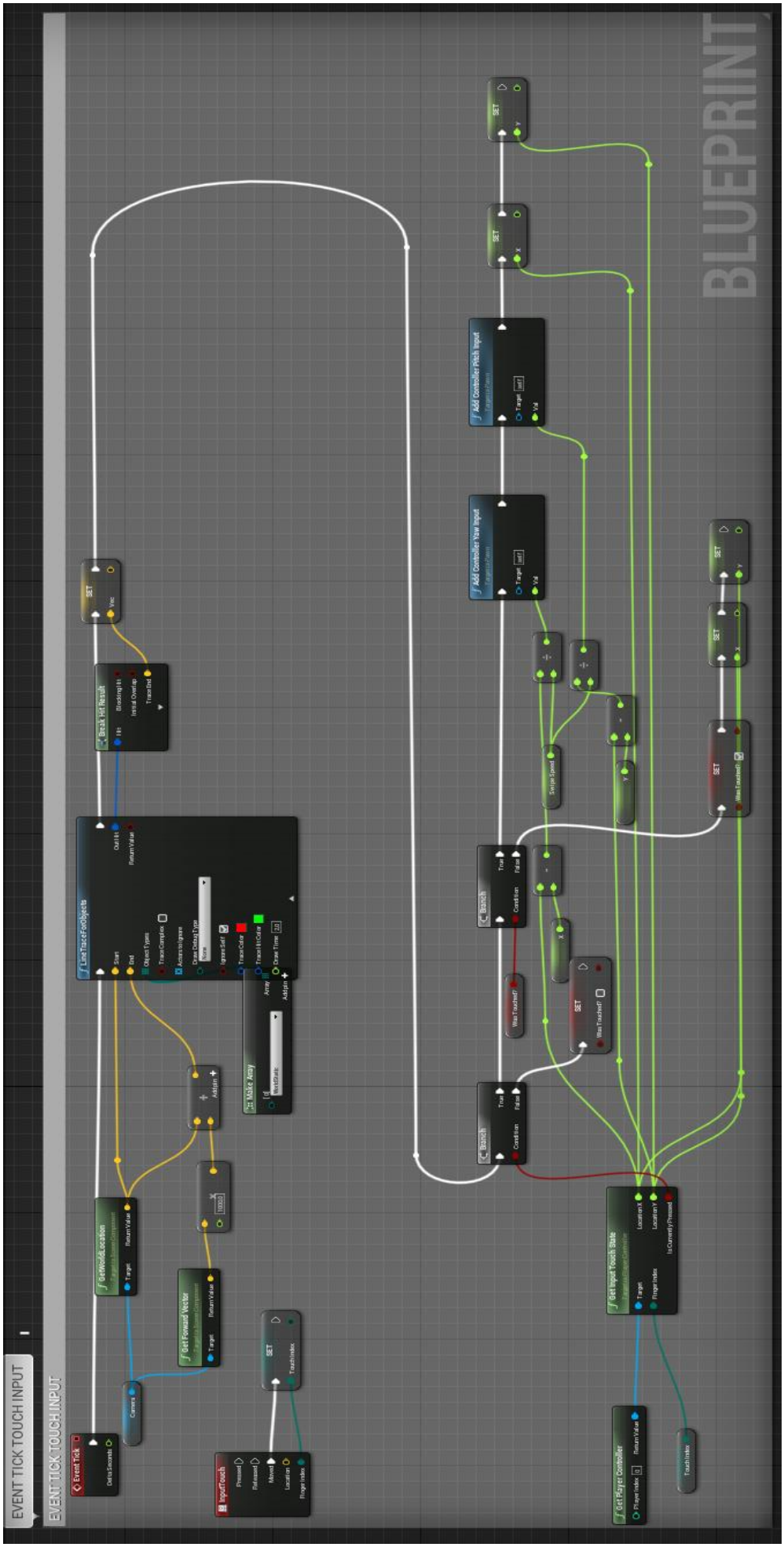


Figure IV-26. Fonction Event Tick.

- Les fonctions dans le bloc **SHOOT WEAPONS** et **ATTAK HAMMER** (figure IV-27) permettent respectivement de faire des attaques avec des bombes et des attaques avec l'arme principale (voir annexe 1 et annexe 2).

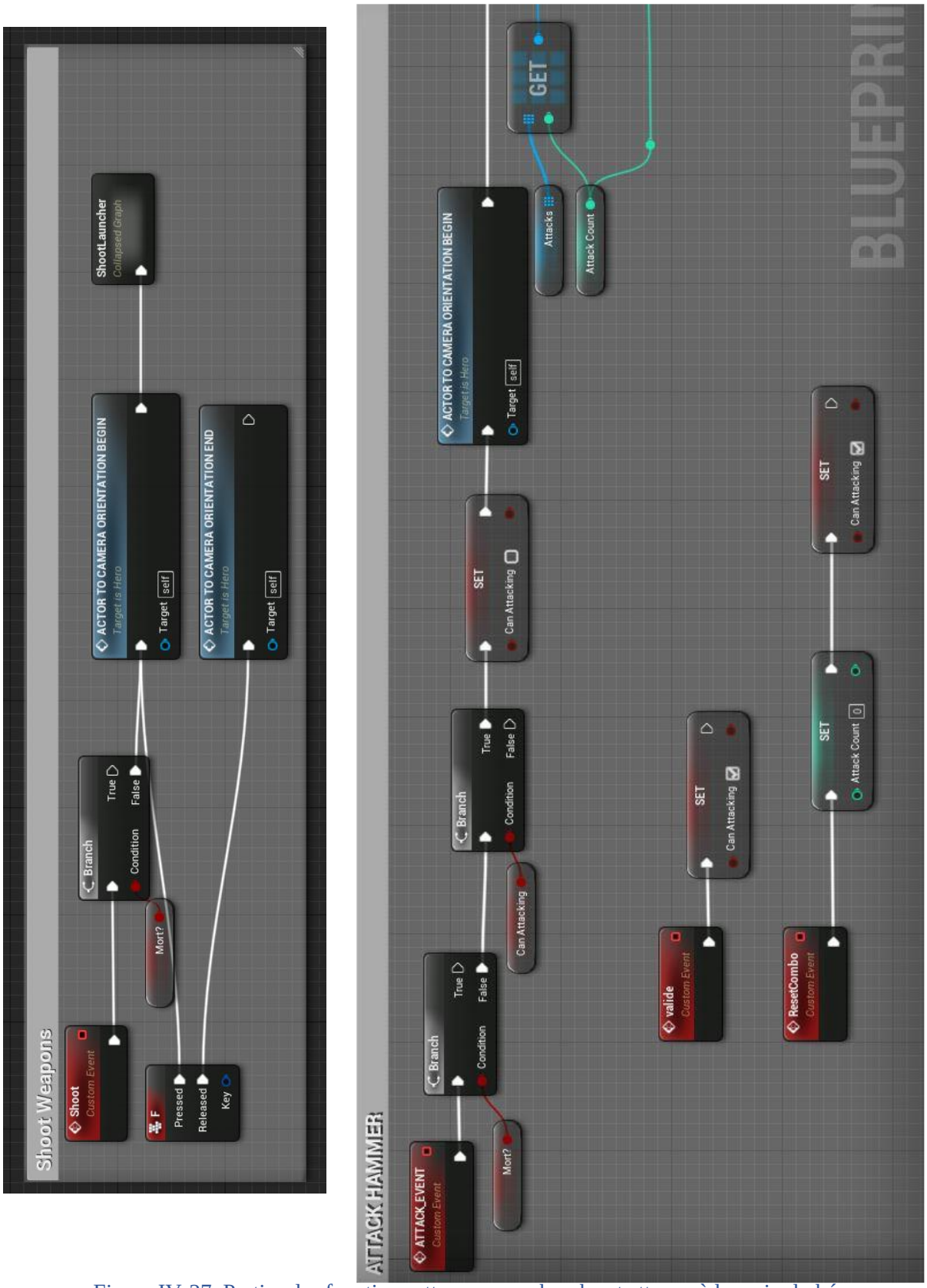


Figure IV-27. Parties des fonctions attaquer avec bombe et attaque à la main du héros.

- La fonction **Event Beginplay** (figure IV-28) est exécutée à chaque initialisation du caractère, elle permet dans notre cas d'initialiser la caméra, d'attacher l'arme à la main du caractère et d'attacher les vertement choisis au caractère (voir annexe 3).

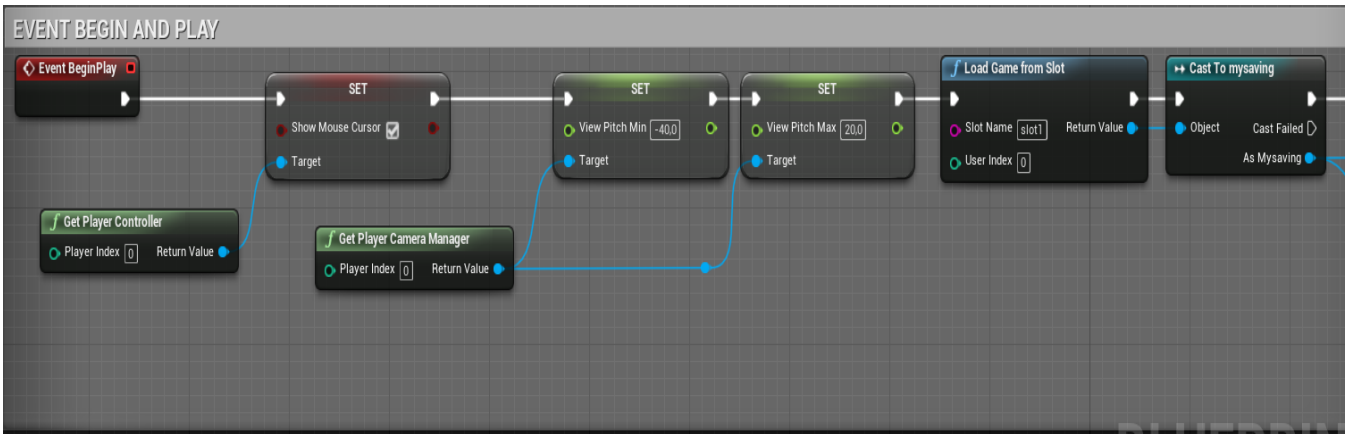


Figure IV-28. Partie du blueprint Event BeginPlay.

- Les fonctions dans le bloc **Take damage And Die** (figure IV-29) sont exécutées quand le joueur subit des dégâts venant de ses ennemis et soustrait les points de dégâts au point de vie. Si les points de vie du joueur atteignent le zéro le script exécute soit **Death Event** si le nombre de vie est supérieur à zéro ou simplement affiche un menu de fin de partie 'GAME OVER'.

- La fonction **Event Any Damage** (figure IV-29) est appelée dans le cas où le joueur subit des dégâts par une bombe (voir annexe 4).

- La fonction **On component Begin Overlap** (figure IV-29) est appelée dans le cas où le joueur subit des dégâts par l'arme principale de l'ennemi (voir annexe 4).

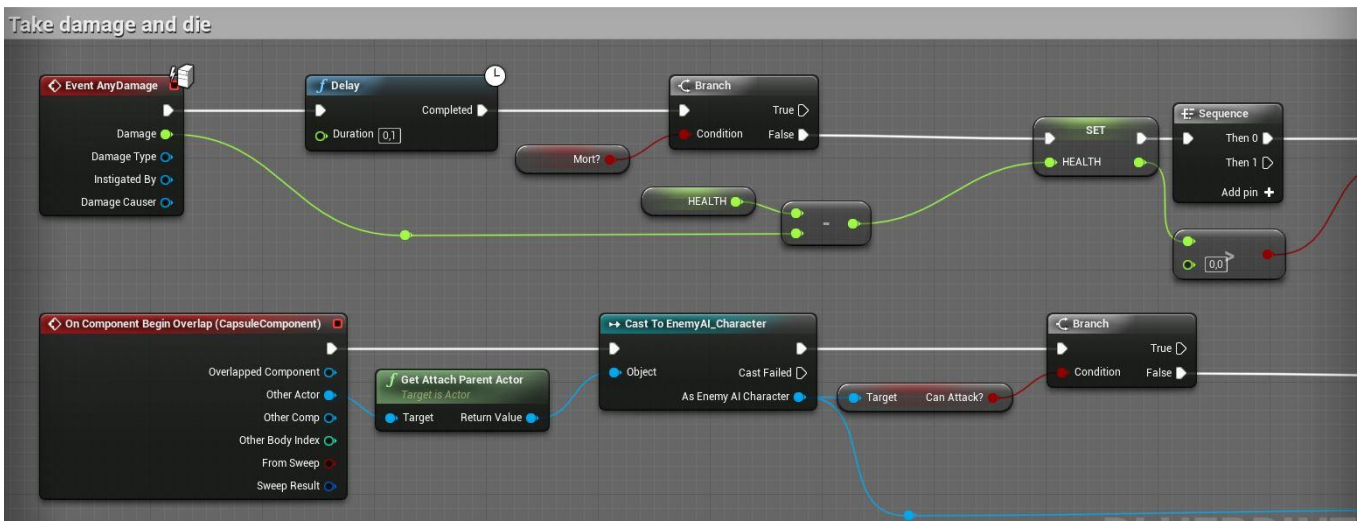


Figure IV-29. Partie des blueprints Take Damage And Die.

- La fonction **Death Event** (figure IV-30) est appelée à la mort du joueur et permet qu'on à elle d'enlever une vie au joueur et de réintroduire de nouveau le joueur dans la partie (voir annexe 5).

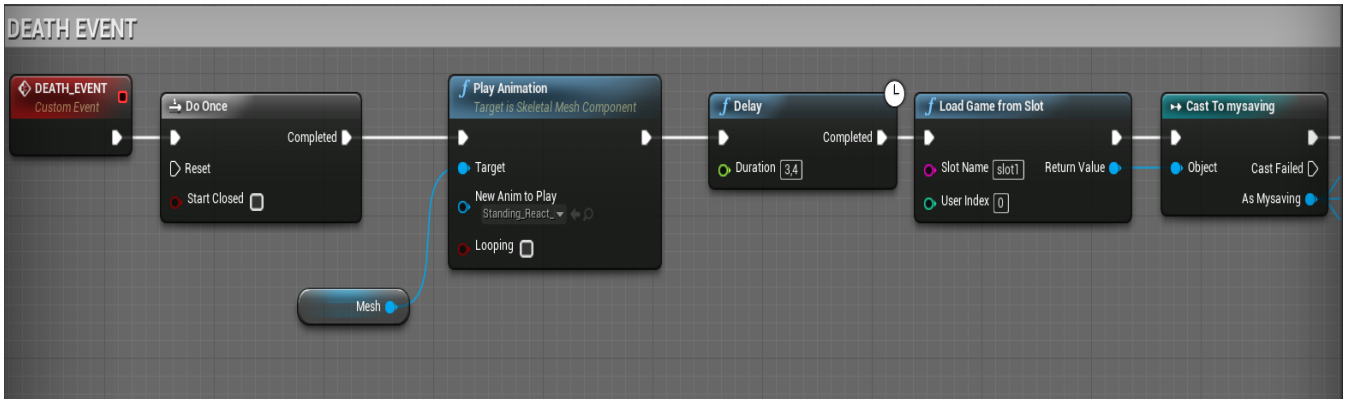


Figure IV-30. Partie de la fonction Death Event.

2. L'ennemi

Le code du caractère ennemi permet à l'intelligence artificielle du moteur d'exécuter des taches comme attaquer le héros, subir des dégâts et changer de comportement en cas de détection du héros.

L'ennemi initialise aussi les variables du graphe de L'AI en faisant appel à ce dernier avec *AIController*.

- Le script d'attaque est semblable à celui du héros (figure IV-31) :

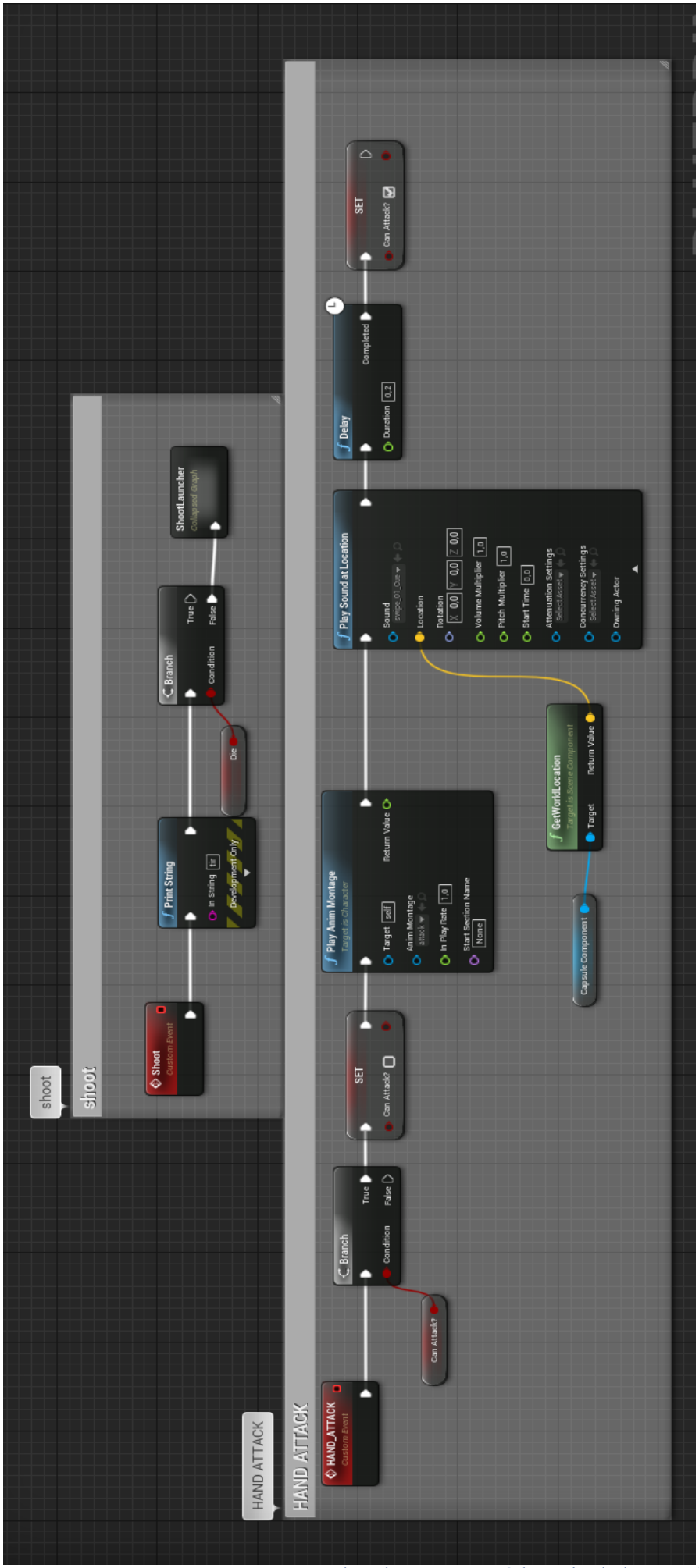


Figure IV-31. Fonctions attaquer avec bombe et attaque à la main de l'ennemi.

- La fonction **Event BeginPlay** (figure IV-32) est exécutée à chaque initialisation du caractère, il permet d'attacher l'arme à la main du caractère et d'initialiser les valeurs du graphe D'AI (voir annexe 6).

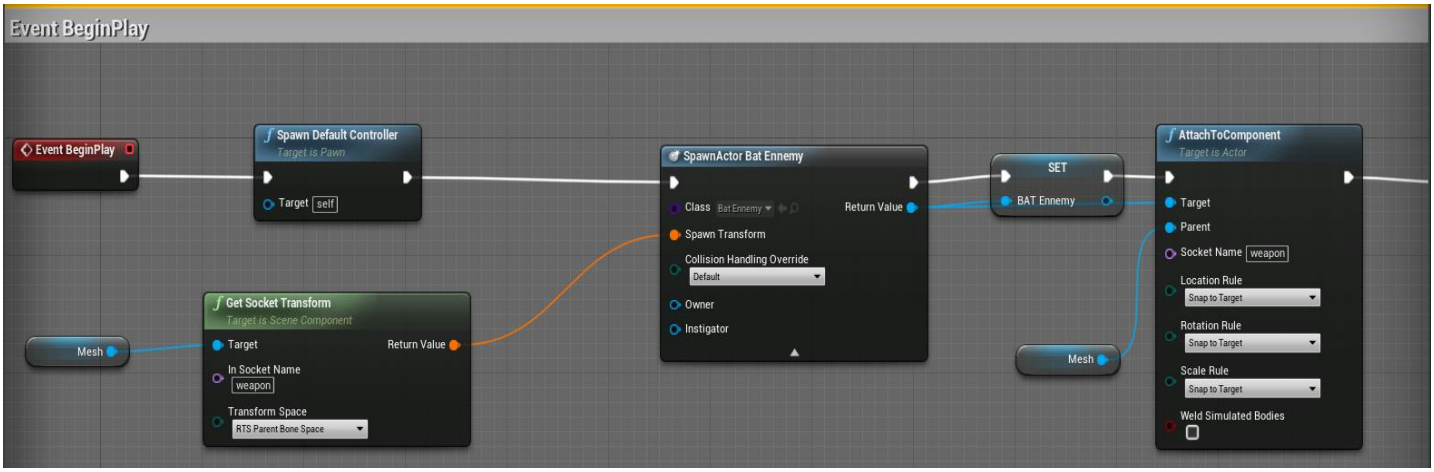


Figure IV-32. Partie de la fonction Event BeginPlay de l'ennemi.

- Les fonctions dans **Take damage And Die** (figure IV-33) sont exécutées quand l'ennemi subit des dégâts venant du héros et soustrait les points de dégâts aux points de vie de l'ennemi. Si les point de vie de l'ennemi atteignent le zéro la fonction **Death Event** s'exécute (voir annexe 7).

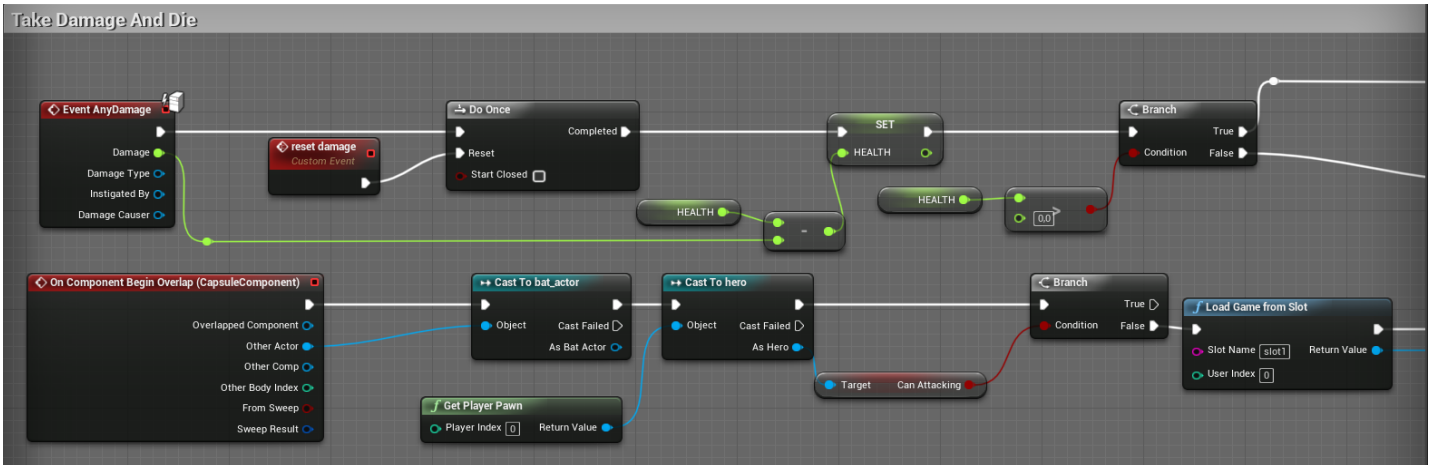


Figure IV-33. Partie du blueprint Take damage And Die de l'ennemi.

La fonction **Death Event** (figure IV-34) est appelée à la mort de l'ennemi dans la partie et permet de jouer l'animation de la mort du personnage, de rajouter du temps, de créer une pièce d'or et enfin d'enlever le caractère de la partie (voir annexe 8).

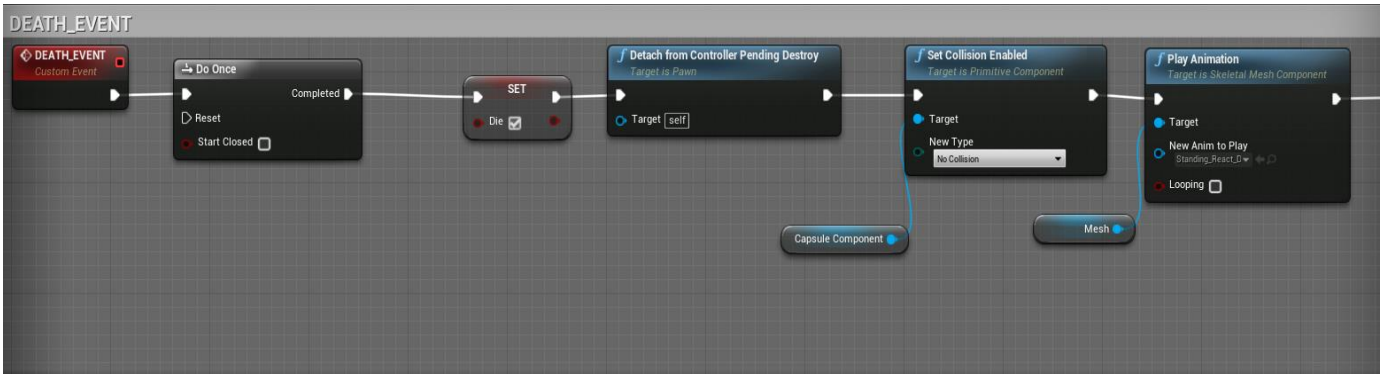


Figure IV-34. Partie de la fonction Death Event de l'ennemi.

- La fonction **OnTargetPerceptionUpdate (AIPerception)** (figure IV-35) qui est dans **Sensing Stimulus** est appelée quand l'ennemi aperçois ou quand il entend le joueur. Elle permet au caractère de changer d'état et d'exécuter d'autres fonctions comme **Event_Start_CheckAwareness_Sight** et **Event_Start_CheckAwareness_Hearing** qui permettent d'augmenter le niveau de suspicion à l'ennemi.

- La fonction **OncomponentBeginOverlap (AITouchPerceptionSphere)** (figure IV-35) est exécutée quand l'ennemi est touché par le joueur et exécute la fonction **AddAwarnessByTouch** (voir annexe 9).

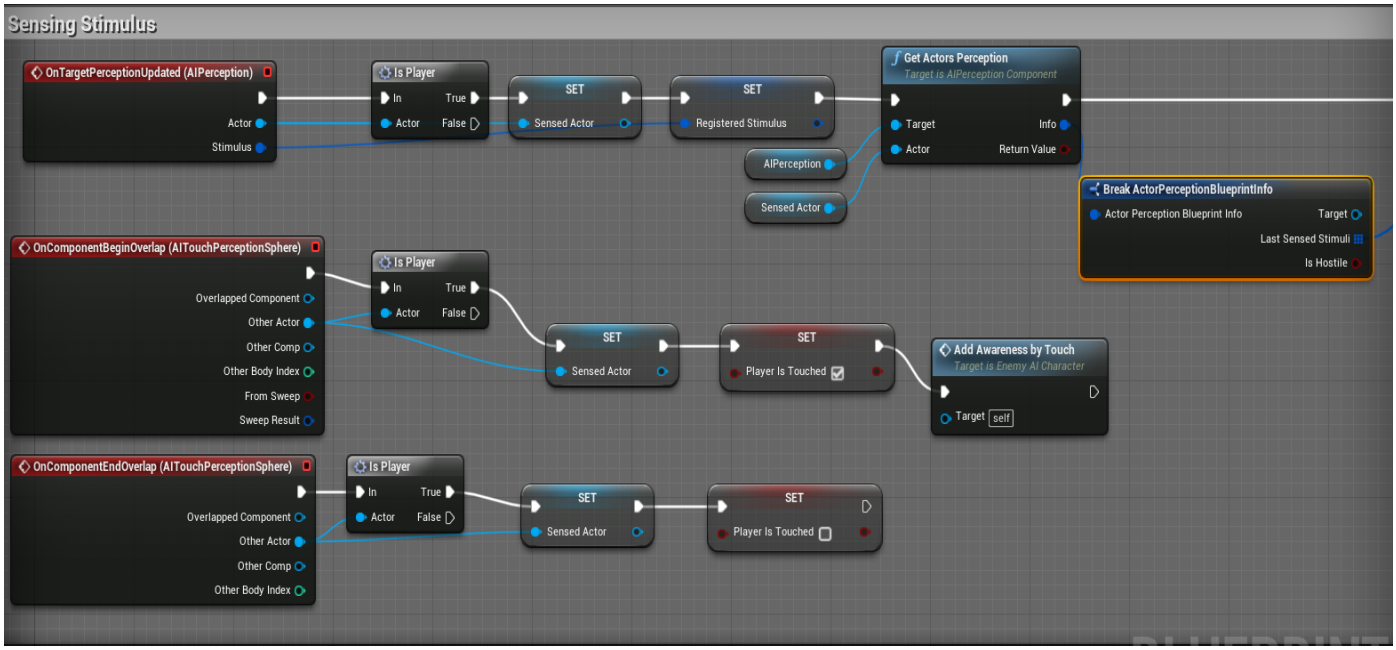


Figure IV-35. Partie du blueprint Sensing Stimulus.

3. Les bombes :

- Les bombes contiennent une simple fonction qui se déclenche à la collision avec un caractère en lui appliquant une perte de point de vie (voir annexe 10).

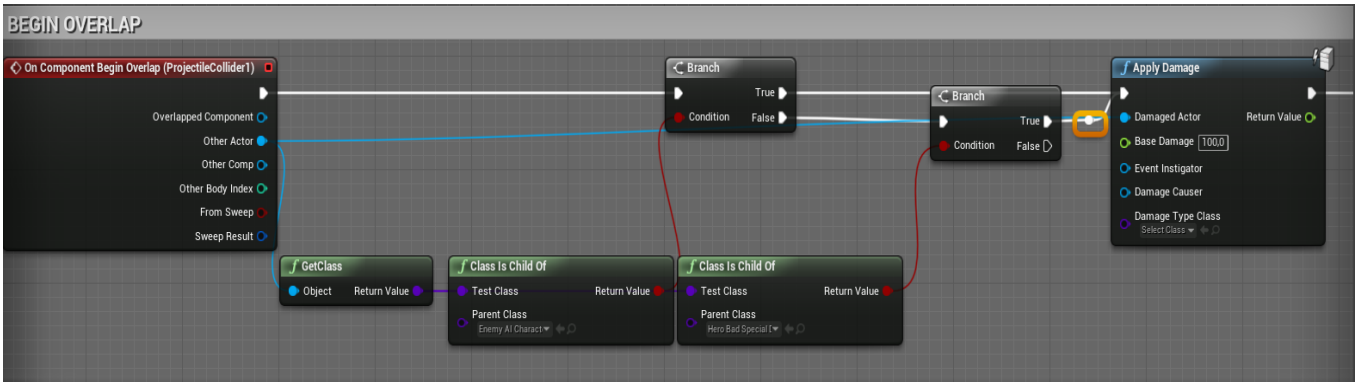


Figure IV-36. Partie de la fonction Begin Overlap d'une bombe.

4. Les bonus

- Les bonus comme les bombes exécutent leurs scripts quand le héros rentre en collision avec le mesh du bonus (**On Component Begin Overlap)**. Un vecteur de probabilité est utilisé pour définir les priorités de sélection d'un bonus.
- Une fonction *'ACTIVE BONUS'* sélectionne un bonus à offrir au joueur (voir annexe 11).

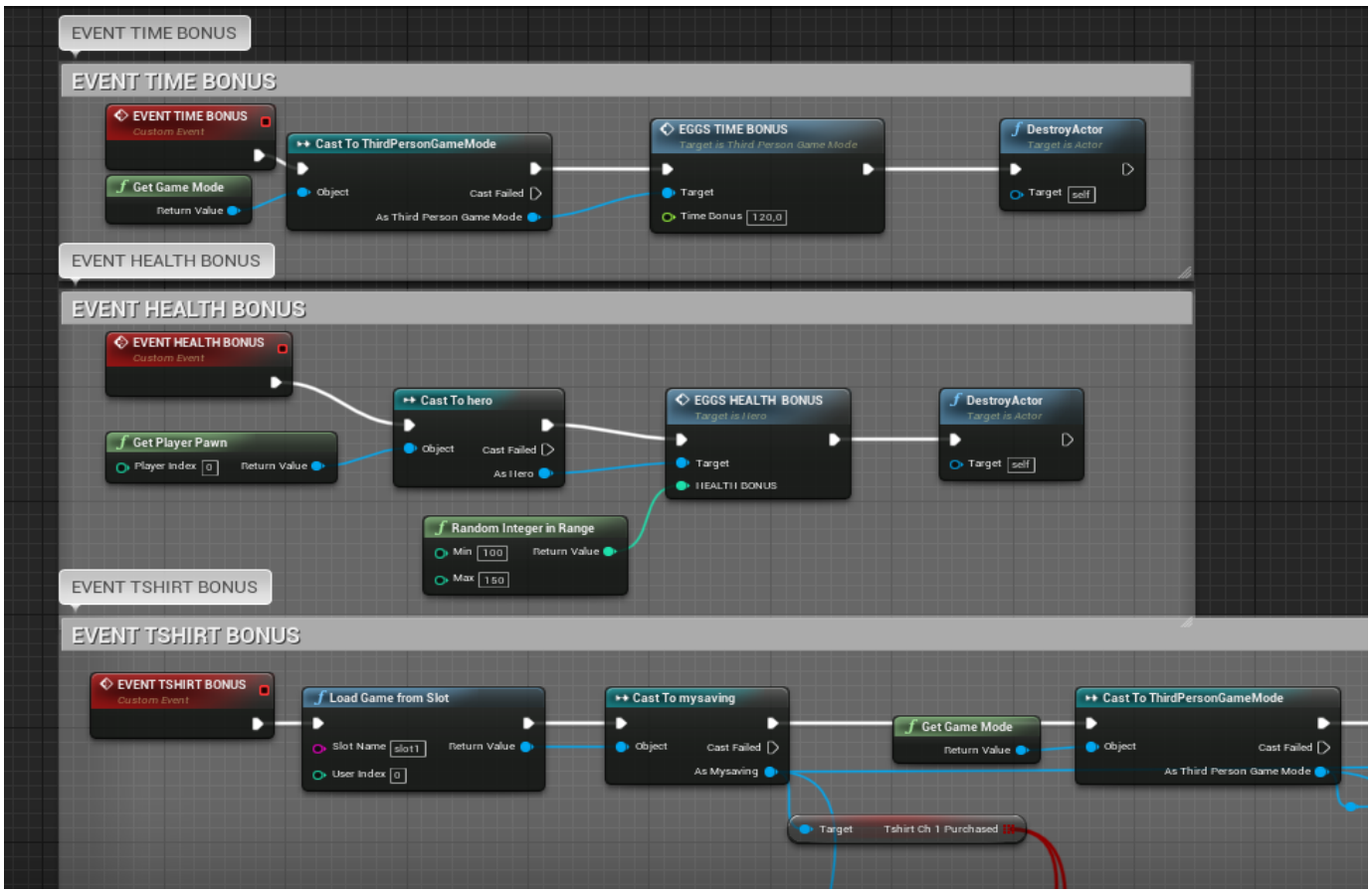


Figure IV-37. Partie du blueprint des œufs bonus.

5. Les pièces

Les pièces que le joueur ramasse lui permettent d'acheter des améliorations dans le menu du héros. Dans la partie le même concept s'applique aux bombes et aux pièces (voir annexe 12).

6. Le Game Mode

Le game Mode permet d'exécuter un code au début d'une partie, dans notre cas le Game Mode nous permet d'avoir en temps réel un chronomètre et de pouvoir lancer dans la partie les ennemis, bombes et bonus (voir annexe 13).

7. Les sauvegardes

Les sauvegardes sont un système de stockage de données sur le disque, elles permettent de stocker de façon permanente les données nécessaires à l'utilisation du jeu.

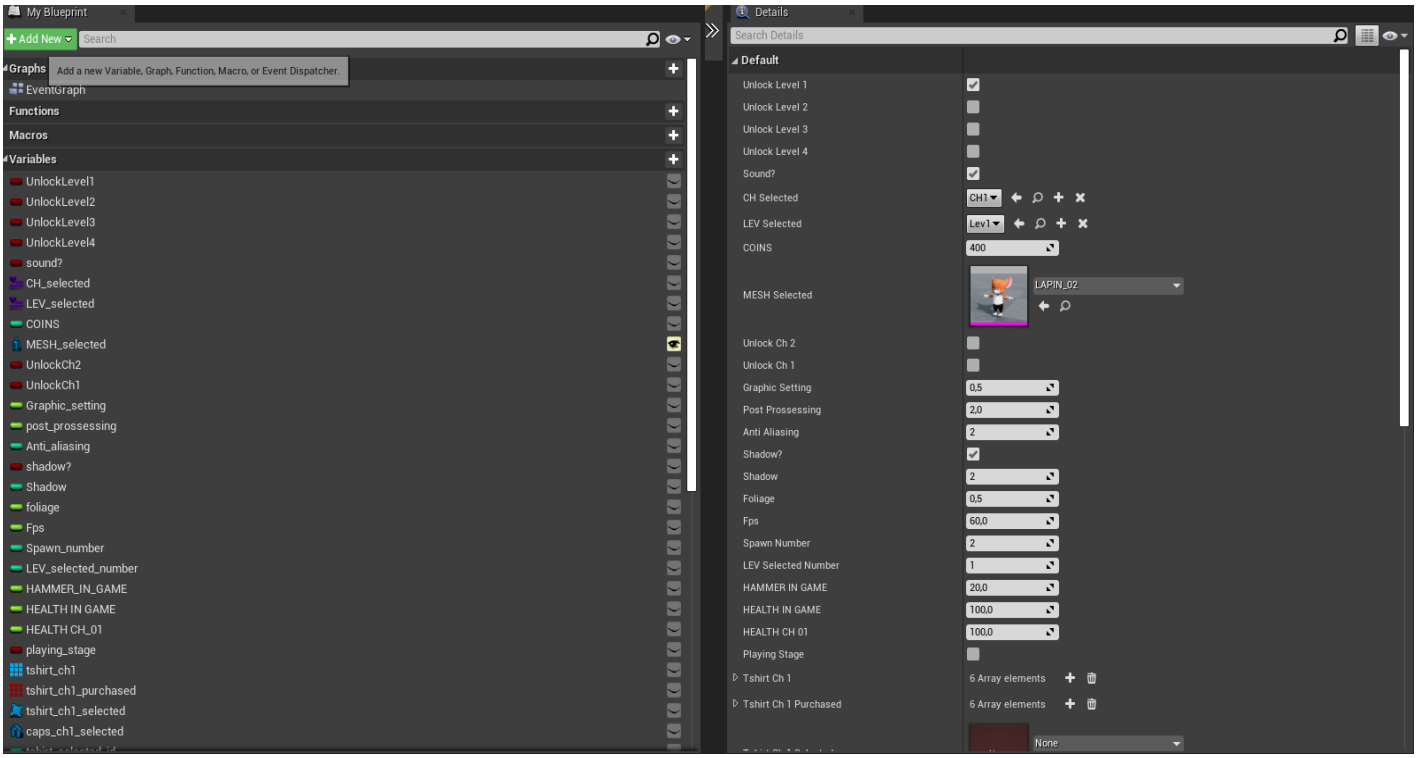


Figure IV-38. Capture d'écran du menu des sauvegardes.

- La création d'une sauvegarde se fait grâce à la fonction **Create Save Game Object**.
- Les fonctions **load Game from Slot** et **Save Game To Slot** permettent de récupérer et de sauvegarder les données de cette sauvegarde partout dans les Blueprint du projet.

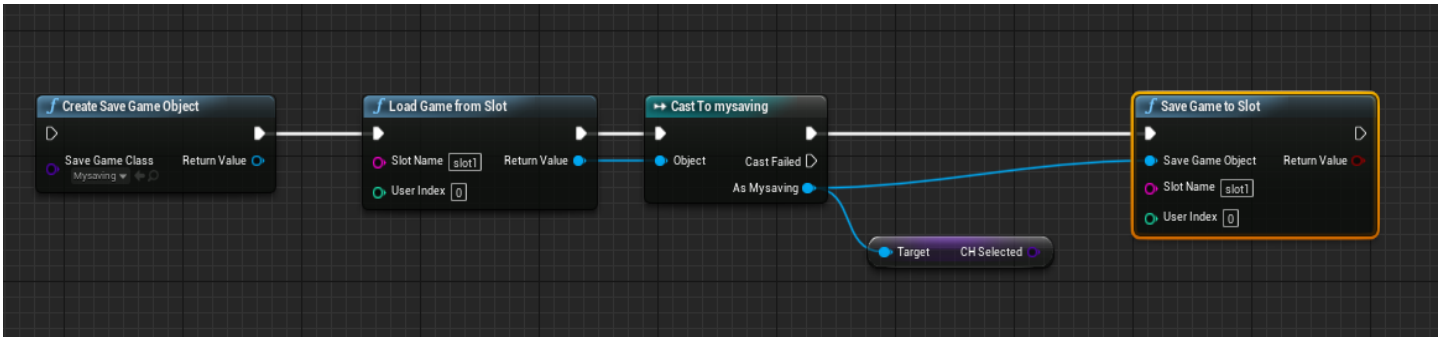


Figure IV-39. Différents nœuds de gestions de sauvegarde.

4. L'intelligence artificielle

Les ennemis gérés par l'intelligence artificielle dans EGGS ADVENTURE ont pour comportement de base une marche suivant des parcours générés par une fonction aléatoire.

Les ennemis peuvent passer par différents niveaux de suspicion allant du niveau 0 qui est la marche au niveau 3 qui est l'hostilité passant par le niveau 1 qui est la suspicion et le niveau 2 qui est la recherche.

Figure IV-40. Système de perception dans EGGS ADVENTURE.

- La partie du code **Sensing Stimulus** déjà vu dans la partie Script des classes est appelée quand l'ennemi aperçoit ou entend le joueur ou qu'il est touché par ce dernier.

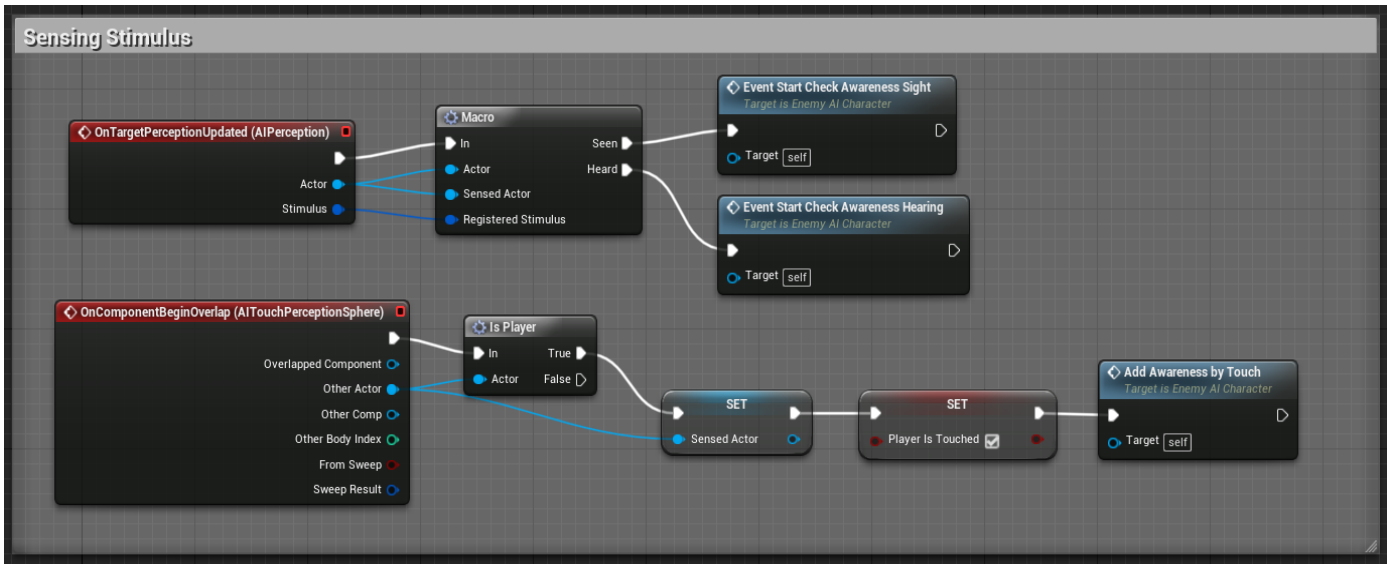


Figure IV-41. Blueprint des différents stimulus dans EGGS ADVENTURE.

- Si le joueur est vu, la fonction **Event_Start_CheckAwareness_Sight** (figure IV-42) est exécutée.
- Si le joueur est entendu, la fonction **Event_Start_CheckAwareness_Hearing** (figure IV-42) est exécutée.
- Ces deux fonctions exécutent deux minuteurs qui font office de boucles qui s'exécutent à chaque période définie dans le champ *Time*, et tant que la condition 'joueur est vu' ou 'joueur est entendu' sont vraies, **Add Awarness by Sight** (figure IV-43) et **Add Awarness by Hearing** (figure IV-44) sont exécutées.

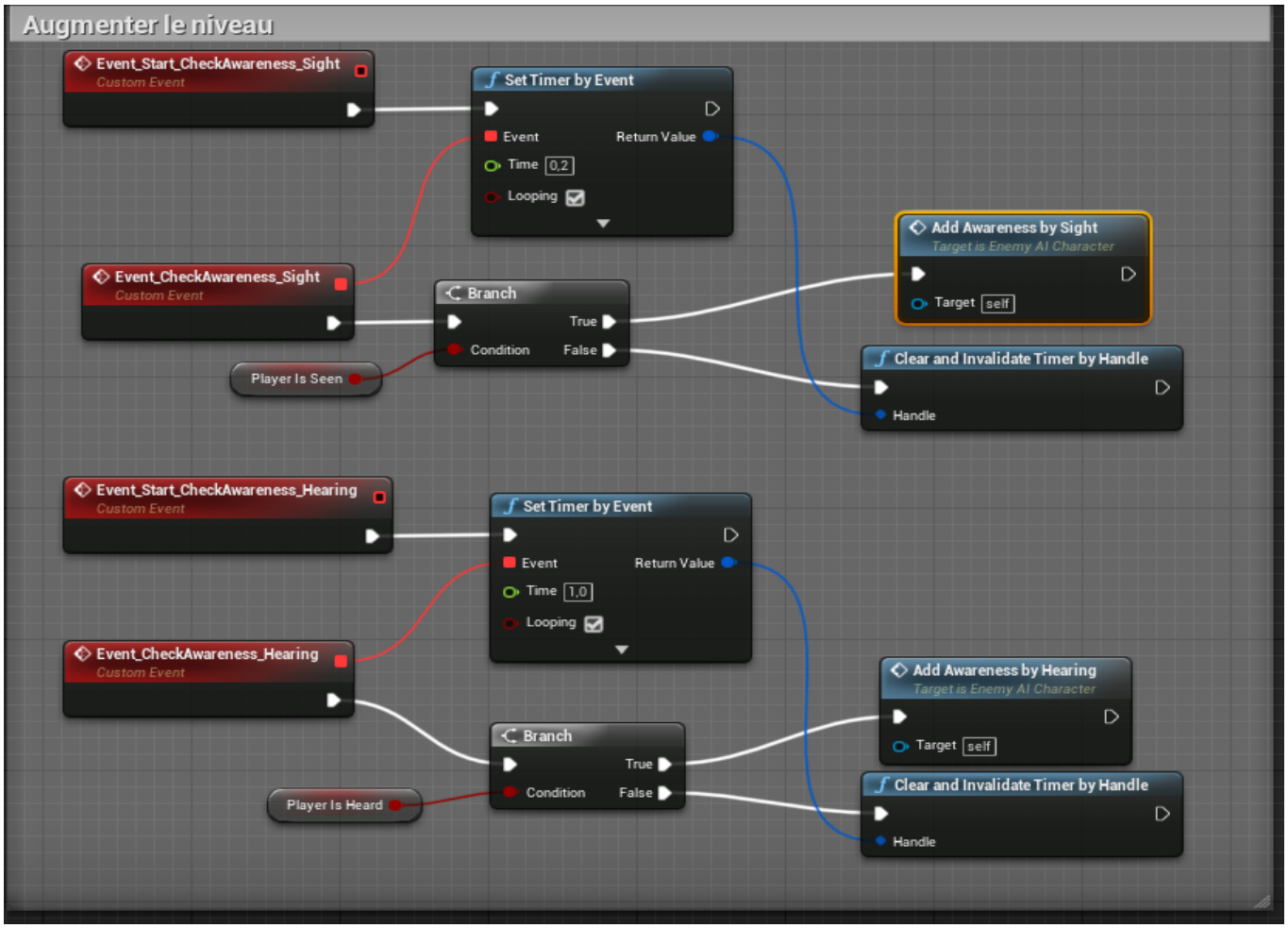


Figure IV-42. Blueprint des fonctions *Event_Start_CheckAwareness_Sight/_Hearing*.

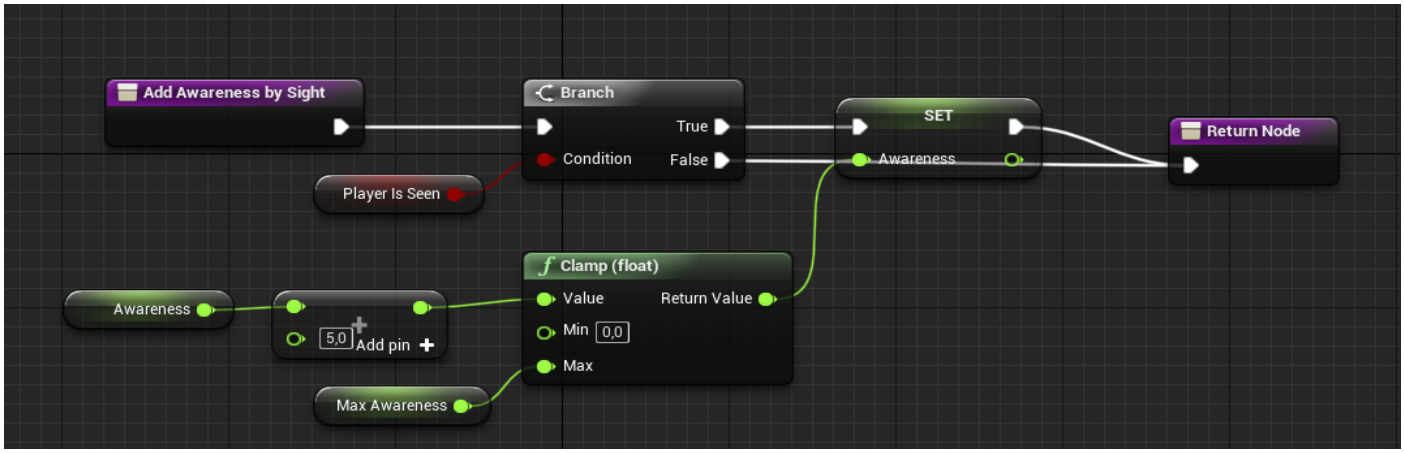


Figure IV-43. Fonction *Add Awarness by Sight*.

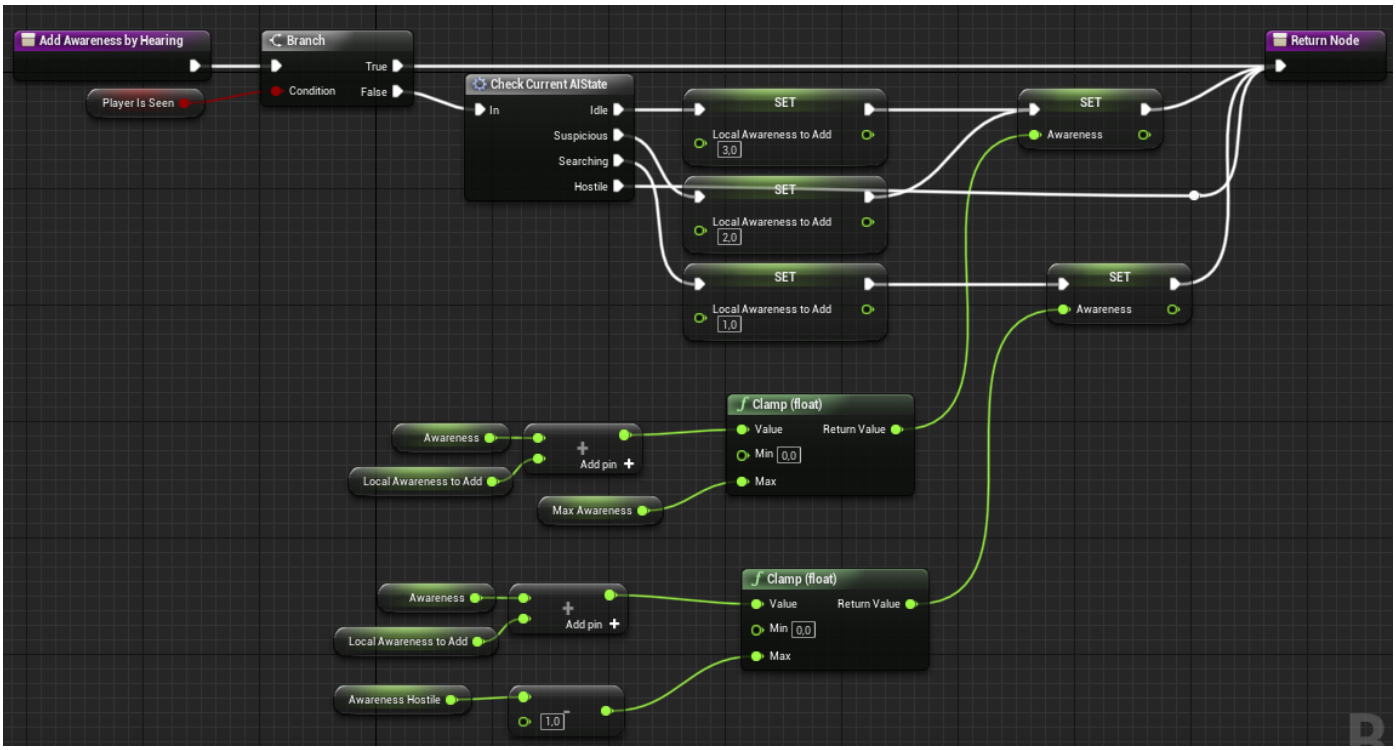


Figure IV-44. Fonction *Add Awarness by Hearing*.

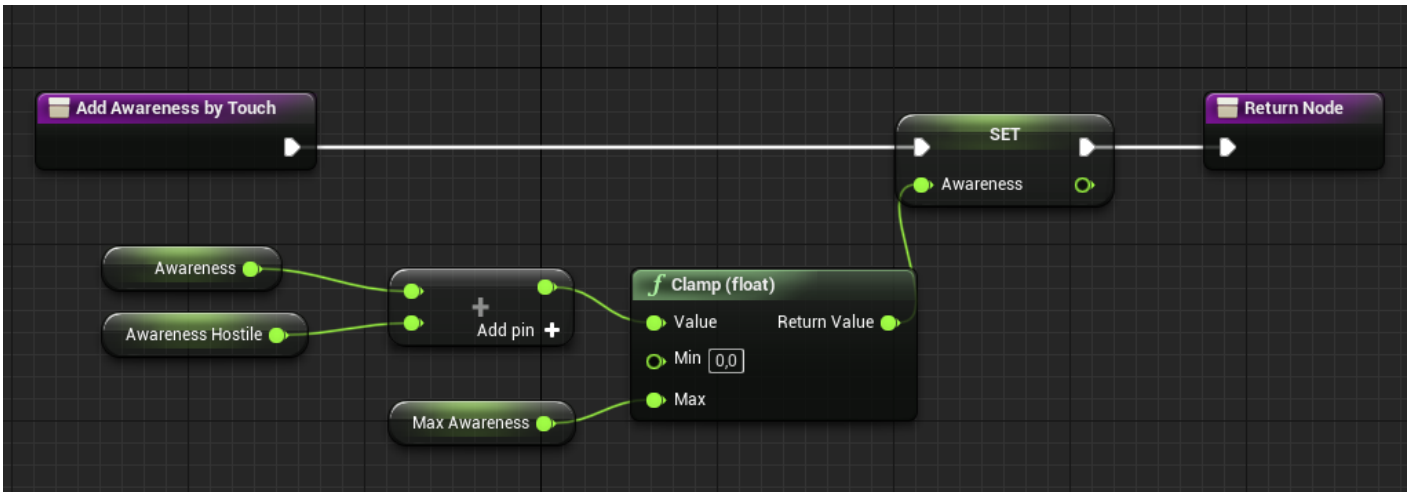


Figure IV-45. Fonction *Add Awarness by Touch*.

- Les trois précédentes fonctions ont pour but principal d'augmenter la variable '*AWARNESS'* qui sert à basculer entre un état et un autre.
- La fonction **Set State by Awarness** (figure IV-47) est exécutée via la fonction **Begin Play** de l'ennemi grâce à un minuteur qui répète la procédure chaque delta temps.

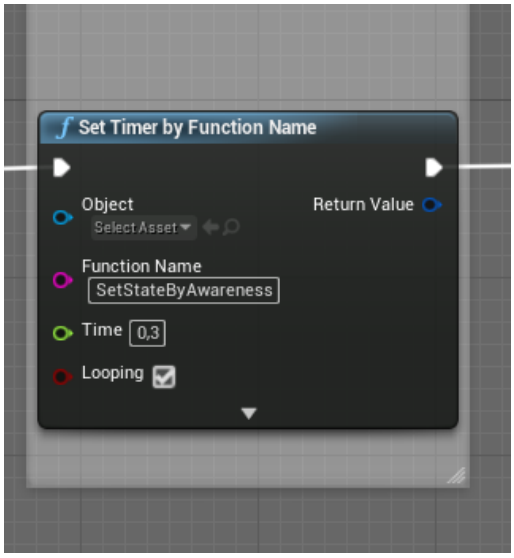


Figure IV-46. Minuteur d'exécution de la fonction *Set State by Awarness*.

- Cette fonction permet d'exécuter la fonction **Transition Enemy AIState** selon la valeur de la variable '*AWARNESSE'*.

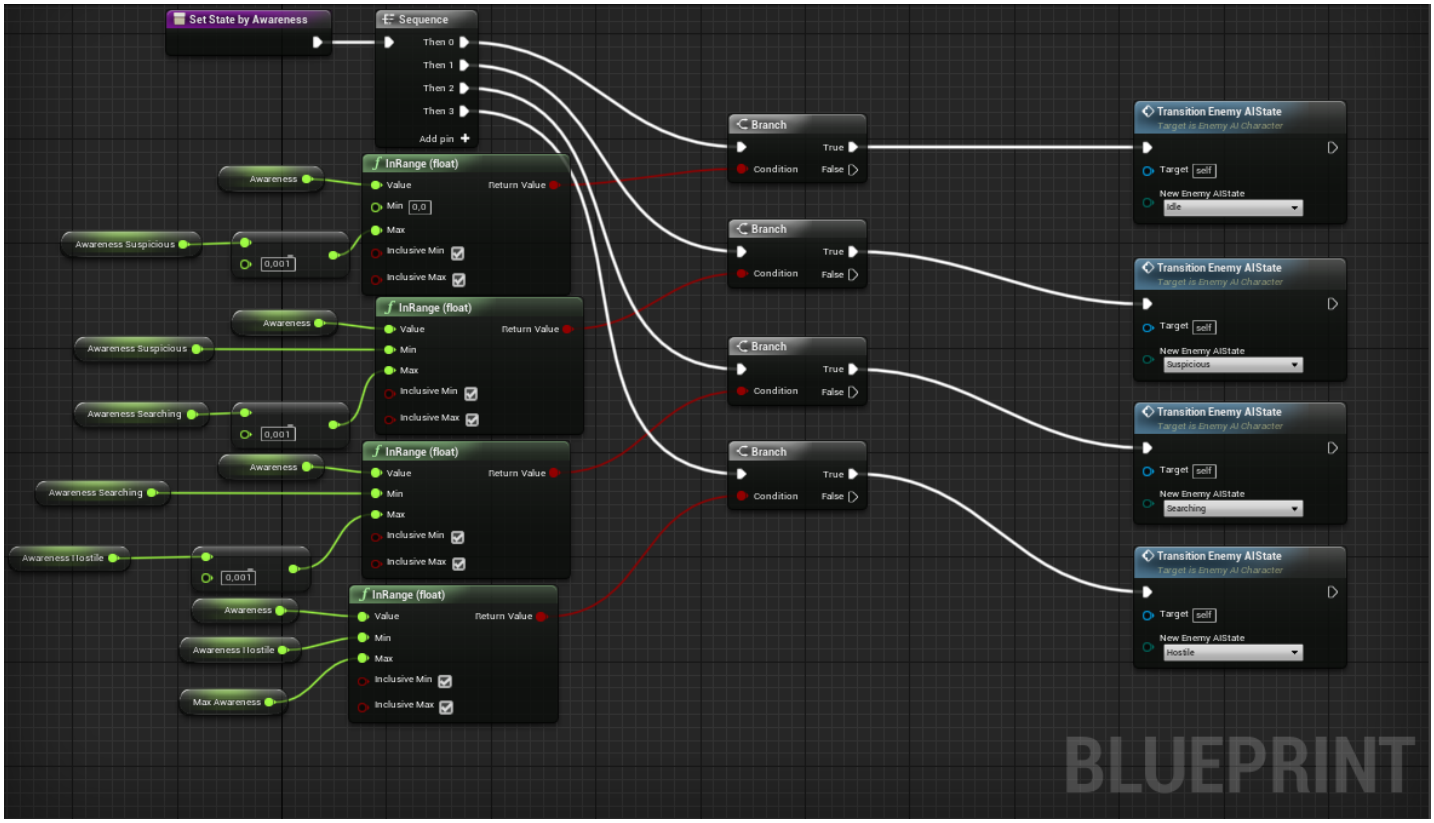


Figure IV-47. Fonction Set State by Awarness.

.

- La fonction **Transition Enemy AIState** (figure IV-48) permet de modifier la valeur du blackBoard '*EnemyAIState'* grâce à laquelle l'arbre de transition change de branche.

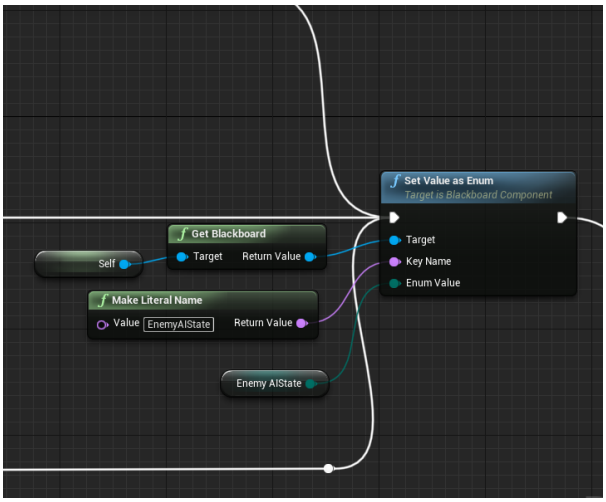


Figure IV-48. Partie de la fonction de modification de la valeur 'EnemyAIState'.

Remarque

Pour faire passer d'un niveau à un niveau inférieur après une période dans laquelle l'ennemi a perdu le joueur, la fonction **CooldownAwareness** permet de diminuer la valeur de la variable AWARNESS.

Cette fonction est exécutée chaque delta temps et vérifie si l'ennemi a perdu le joueur (voir annexe 14).

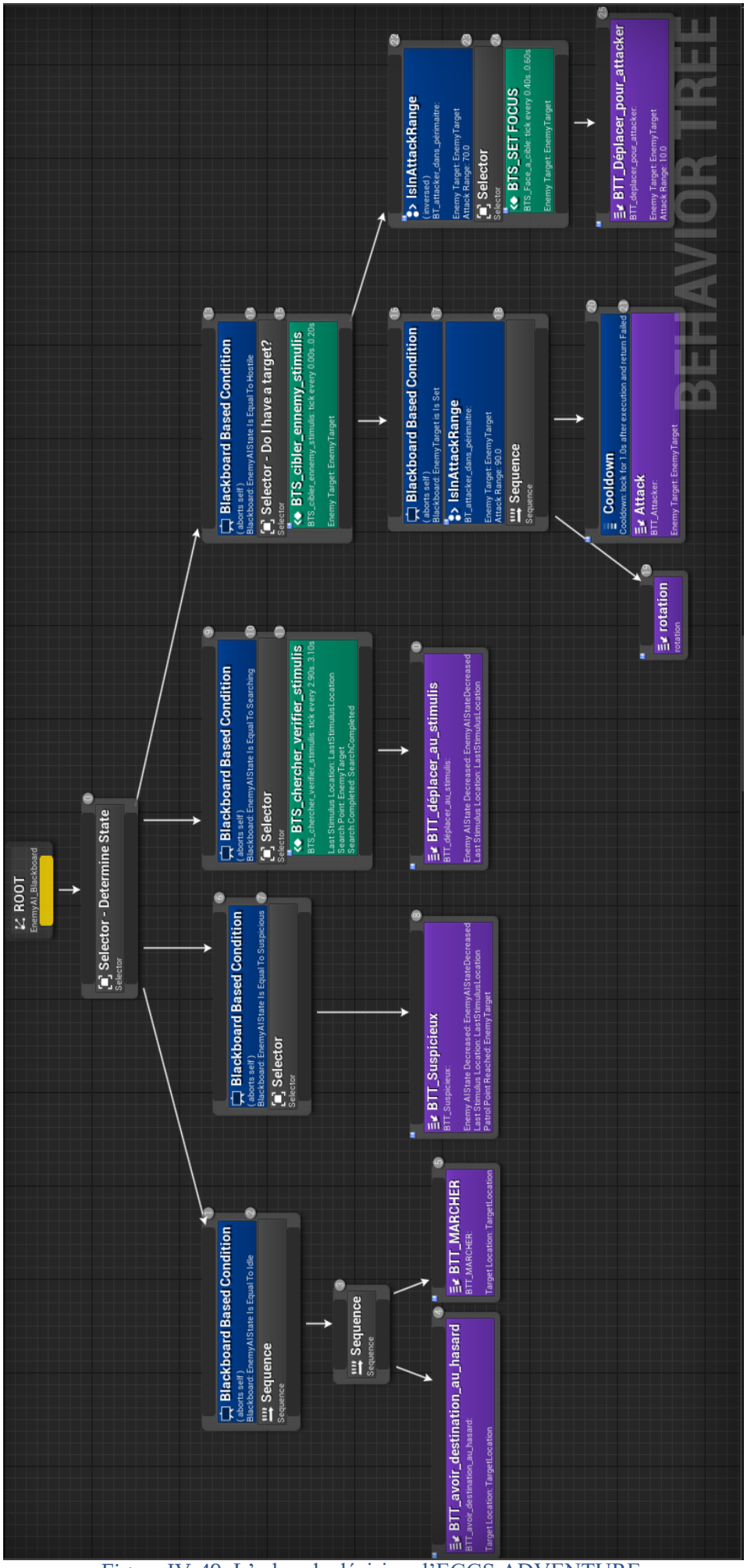


Figure IV-49. L'arbre de décision d'EGGS ADVENTURE.

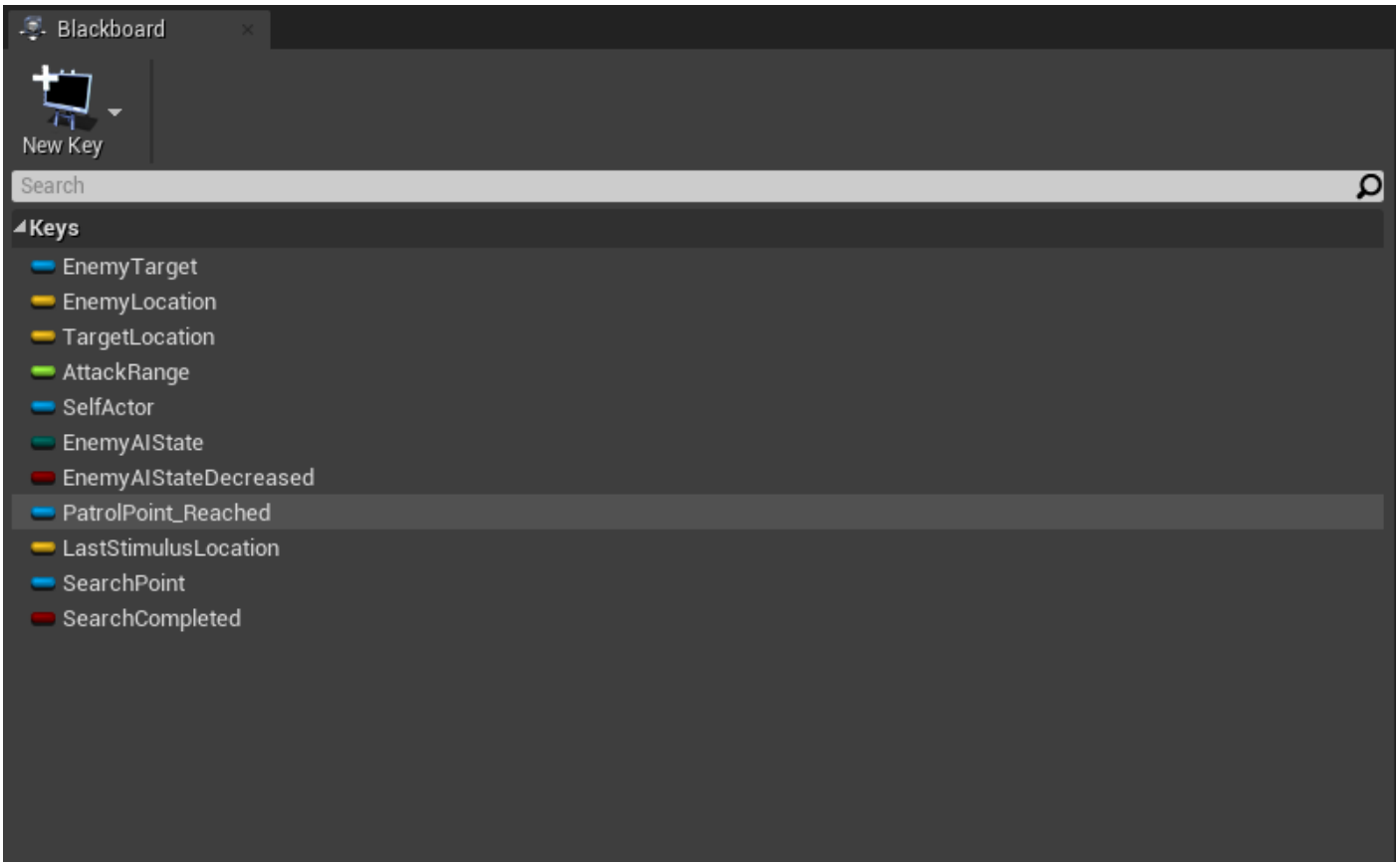


Figure IV-50. Ensemble de clés utilisées dans le BlackBoard d'EGGS ADVENTURE.

- Chaque branche de l'arbre est sélectionnée selon un composite avec un décorateur basé sur la vérification de la clé du blackboard 'EnemyAIState'

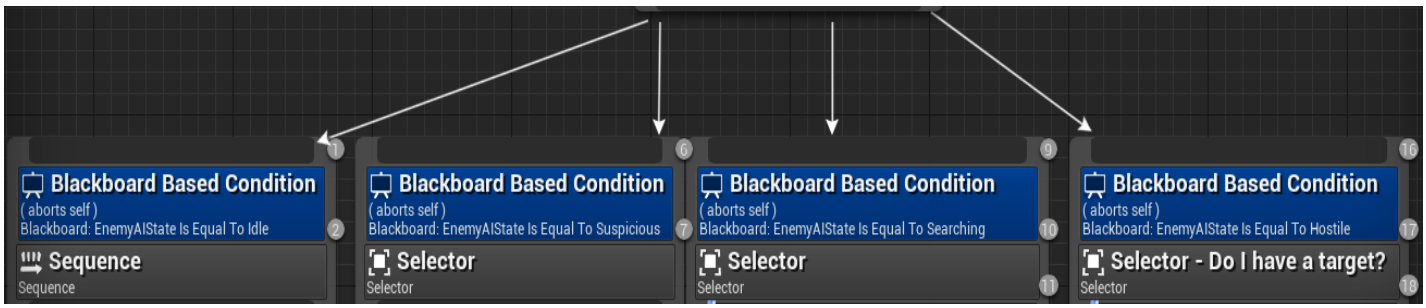


Figure IV-51. Liste des composites avec leurs décorateurs.

- La branche Recherche (figure IV-52) comporte un service 'BTS_chercher_verifier_stimulis' (figure IV-53)qui récupère de l'ennemi le dernier endroit où le joueur a été détecté.
  L'information est enregistrée dans la clé du blackboard 'LastStimulusLocation' afin que l'ennemi puisse se déplacer à l'endroit indiqué et rechercher le joueur dans le cas où l'ennemi a perdu le joueur.

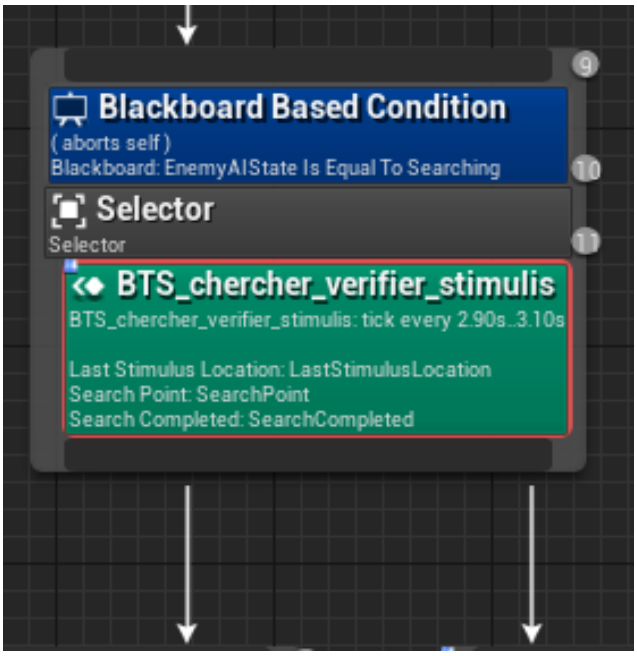


Figure IV-52. Branche Recherche.

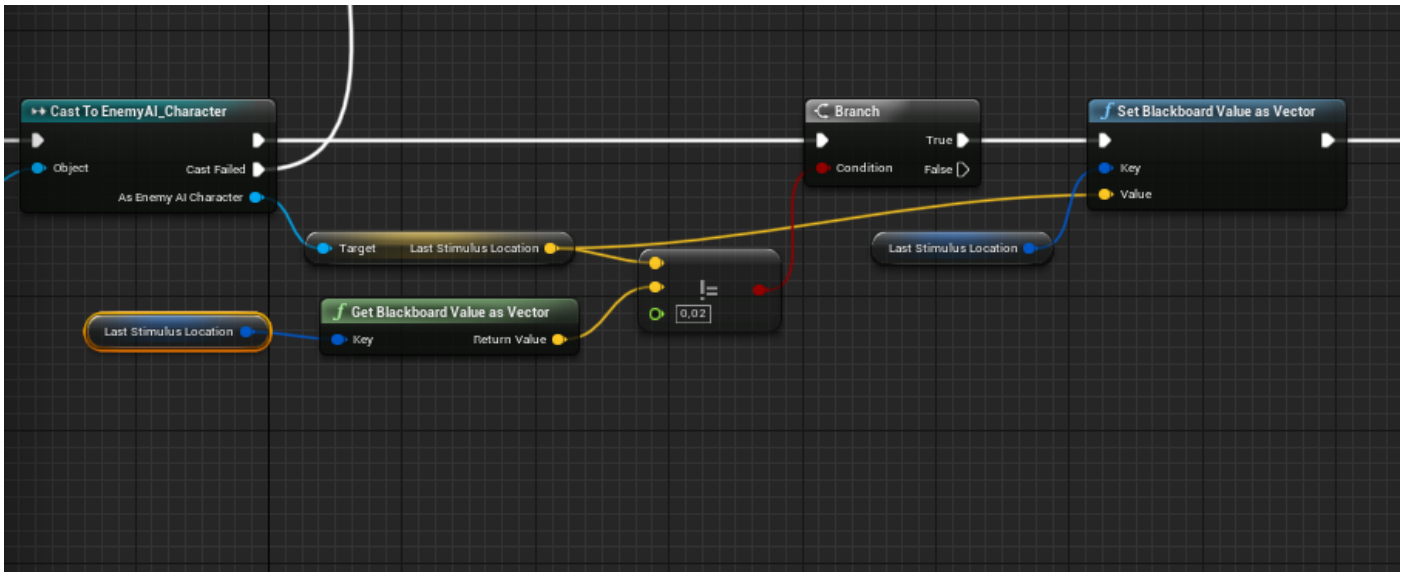


Figure IV-53. Partie du Blueprint ‘BTS_chercher_verifier_stimulis.

- La Branche Hostile (figure IV-54) comporte aussi un service 'BTS_cibler_ennemy_stimulis' (figure IV-55) qui récupère l'identité du joueur détecté par l'ennemi et l'enregistre dans la clé du blackboard 'Enemy Target'.

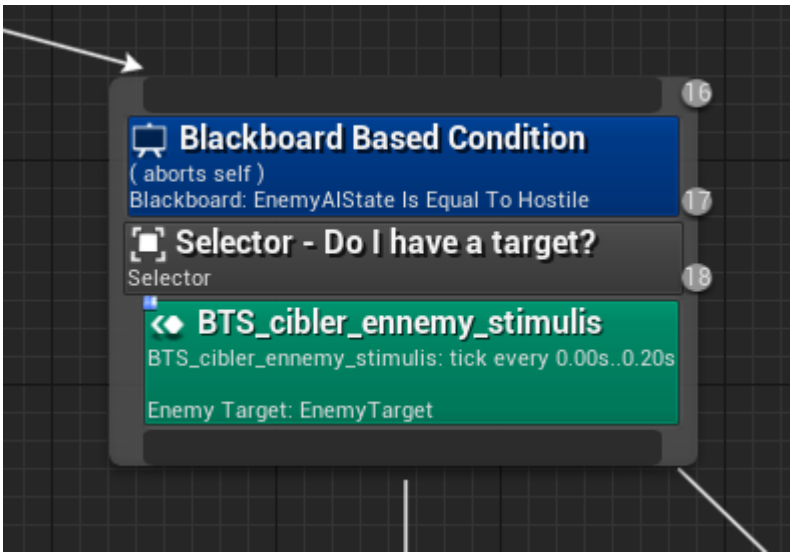


Figure IV-54. Branche Hostile.

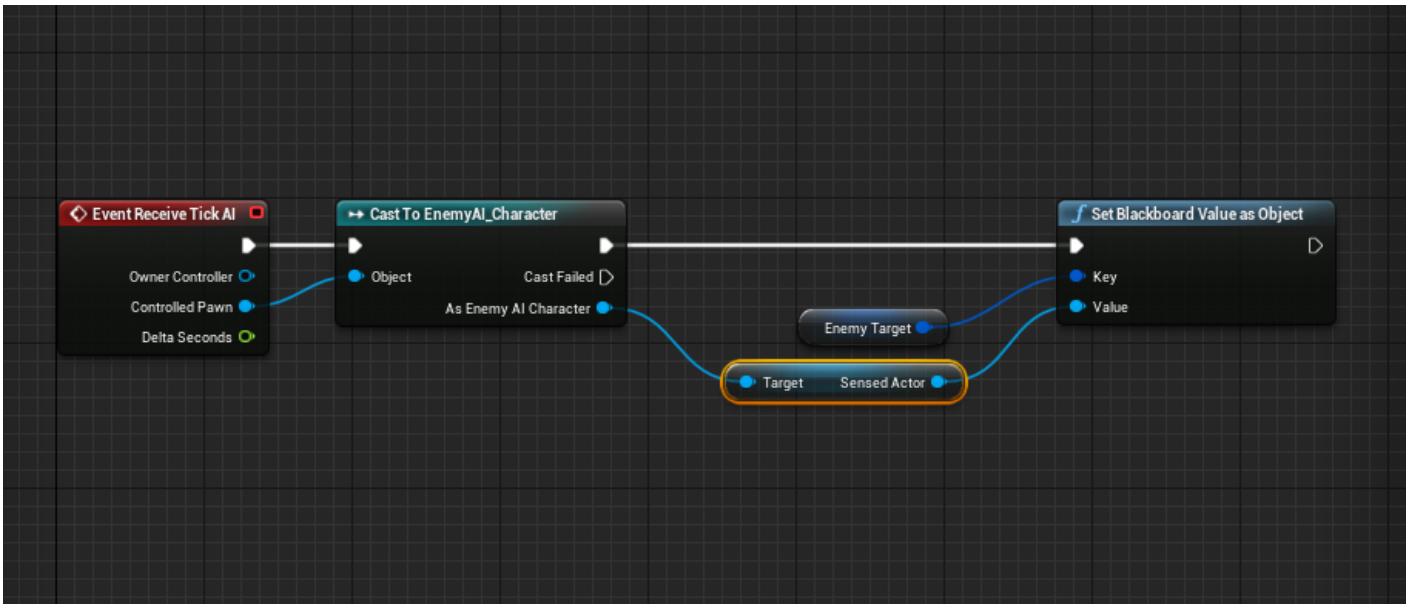


Figure IV-55. Partie de la fonction 'BTS_cibler_ennemy_stimulis.

- La branche Marche comporte deux tâches :
  - 'BTT_avoir_destination_au_hasard' (figure IV-56) : renvoie une destination aléatoire ou l'ennemi devra se déplacer.
  - 'BTT_MARCHER' (figure IV-57) : permet à l'ennemi de se déplacer vers la destination préalablement enregistrée.

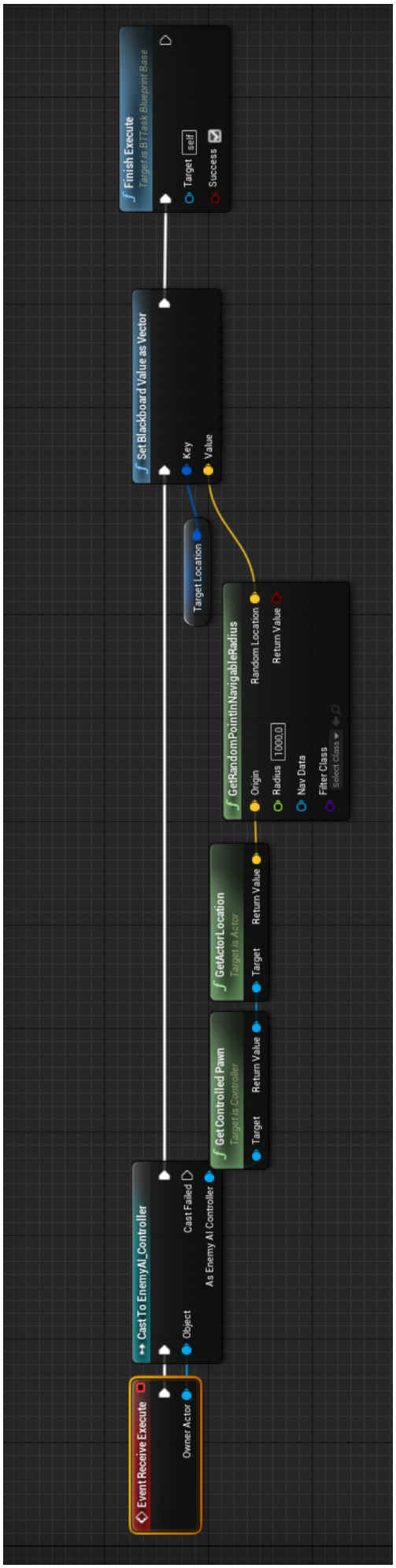


Figure IV-56. Fonction BTT_avoir_destination_au_hasard

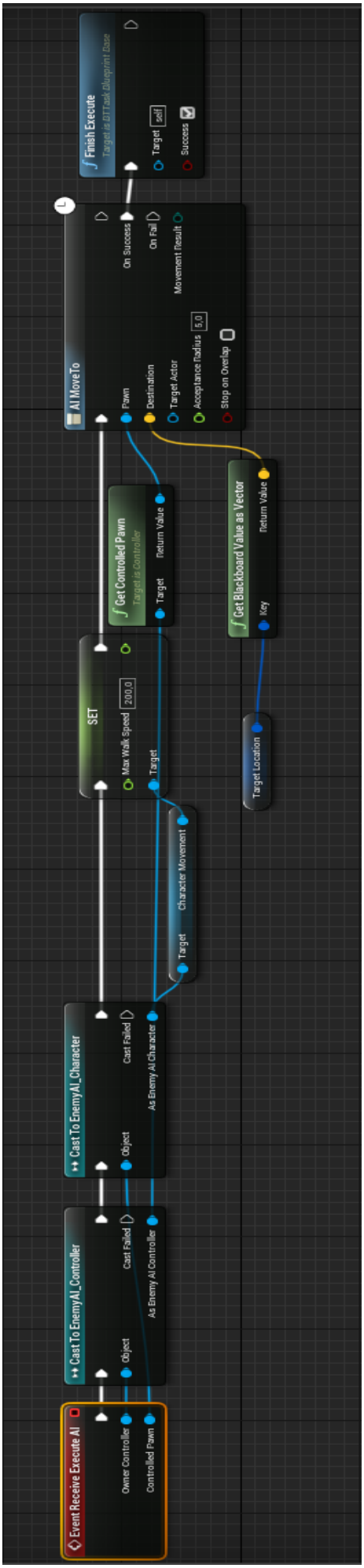


Figure IV-57. Fonction BTT_MARCHER

- La branche Suspicion comporte une seule tâche (figure IV-58) qui permet d'arrêter le mouvement de l'ennemi et de jouer un son.

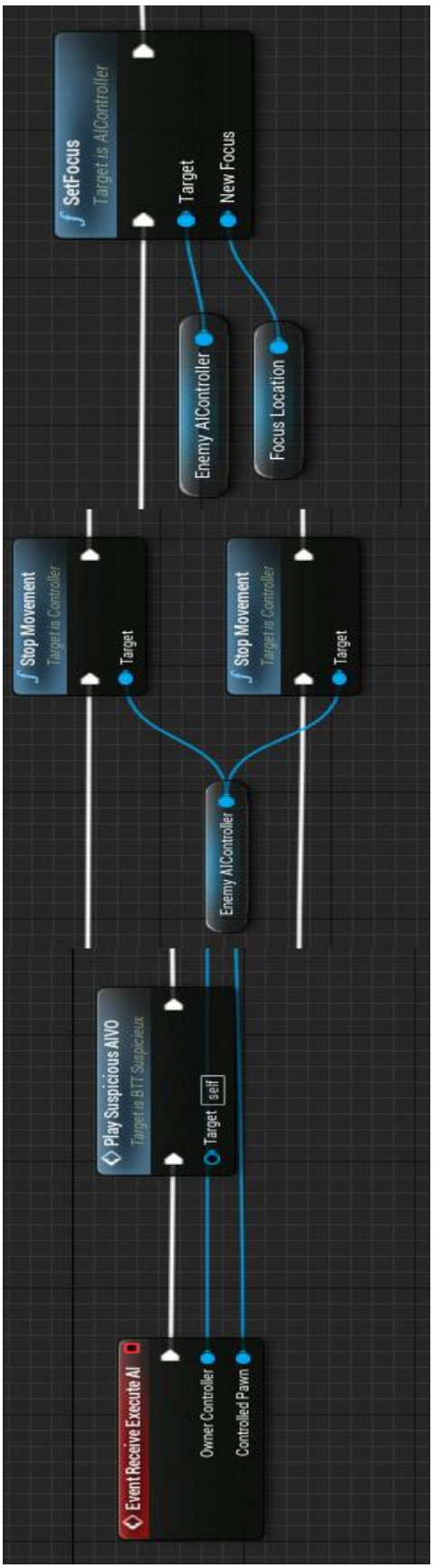


Figure IV-58. Parties du blueprint BTT_Suspicieux.

- La branche Recherche comporte une tâche :
  - ‘BTT_déplacer_au_stimulis’ (figure IV-59) : tâche qui permet à l’ennemi de se déplacer au dernier endroit où le joueur a été signalé.

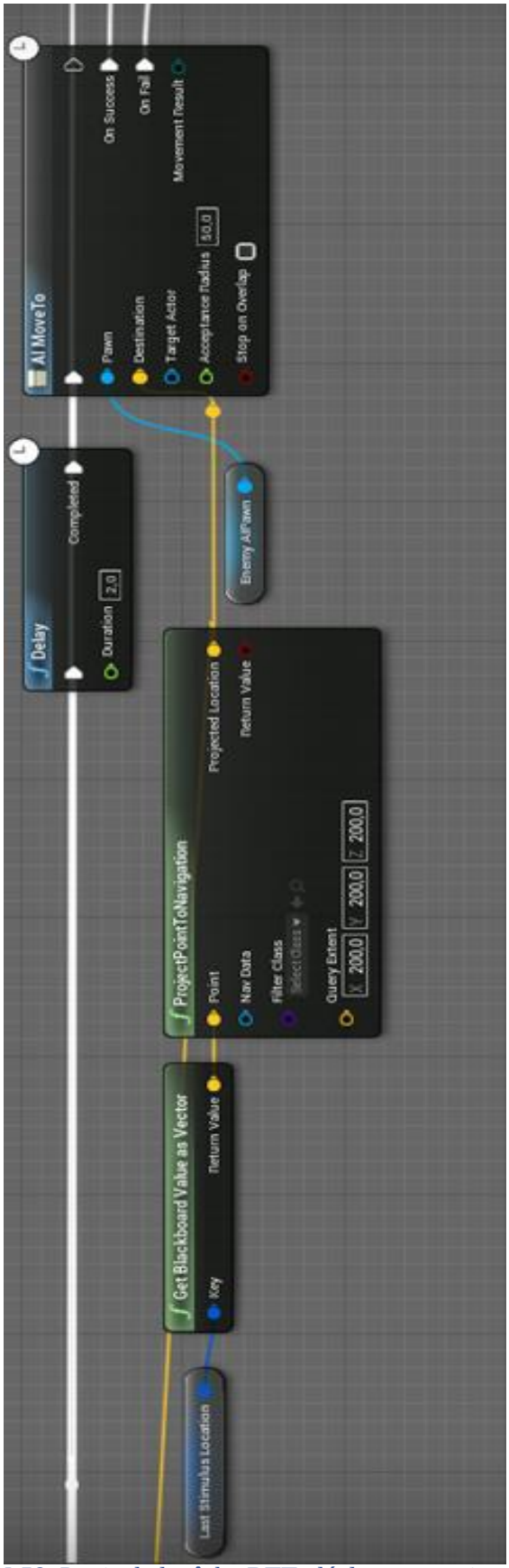


Figure IV-59. Partie de la tâche BTT_déplacer_au_stimulis.

- La branche Hostile (figure IV-60) a deux branches qui avec chaque une un décorateur ‘IsInAttackRange’ qui permet d’exécuter la première branche si la distance entre le joueur et l’ennemi est inférieure à 90 unités et exécute la deuxième branche si la distance est supérieure à 70 unités (figure IV-61).

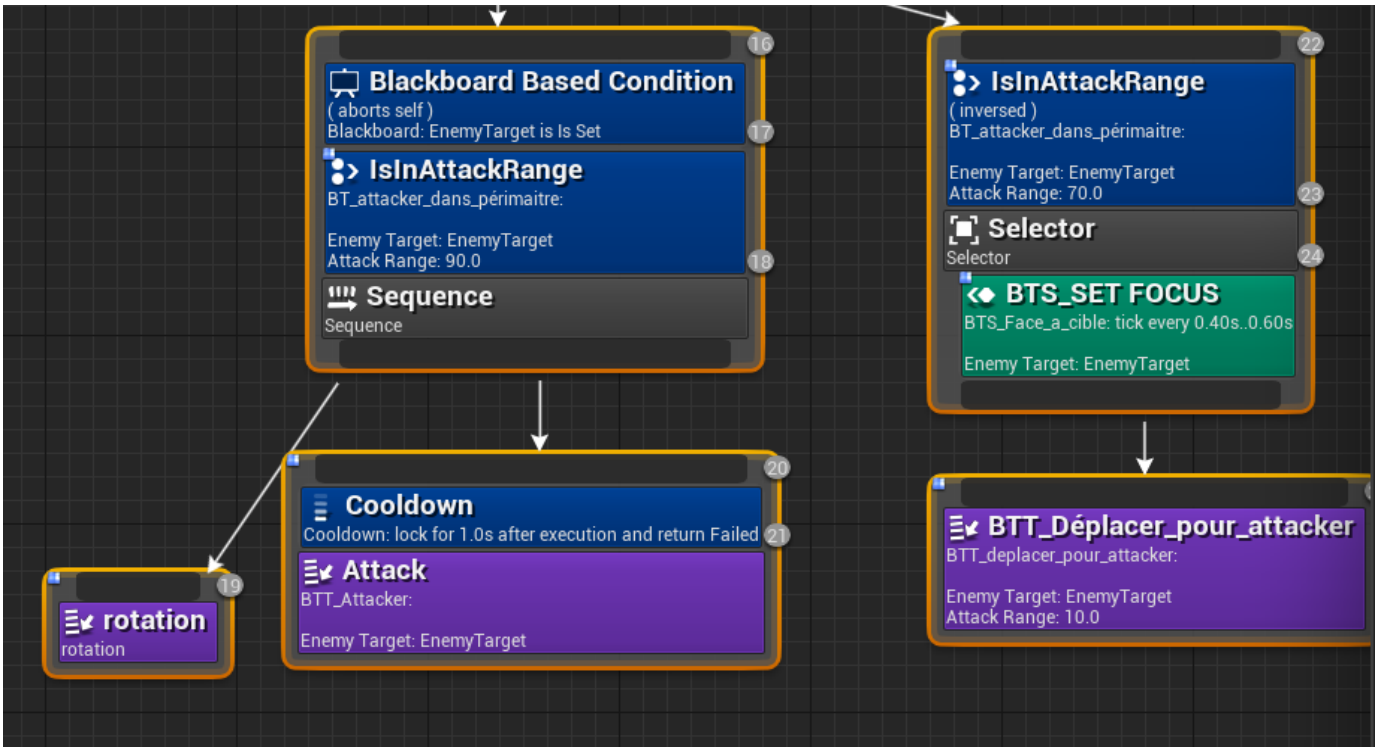


Figure IV-60. Les deux sous branches de la branche Hostile.

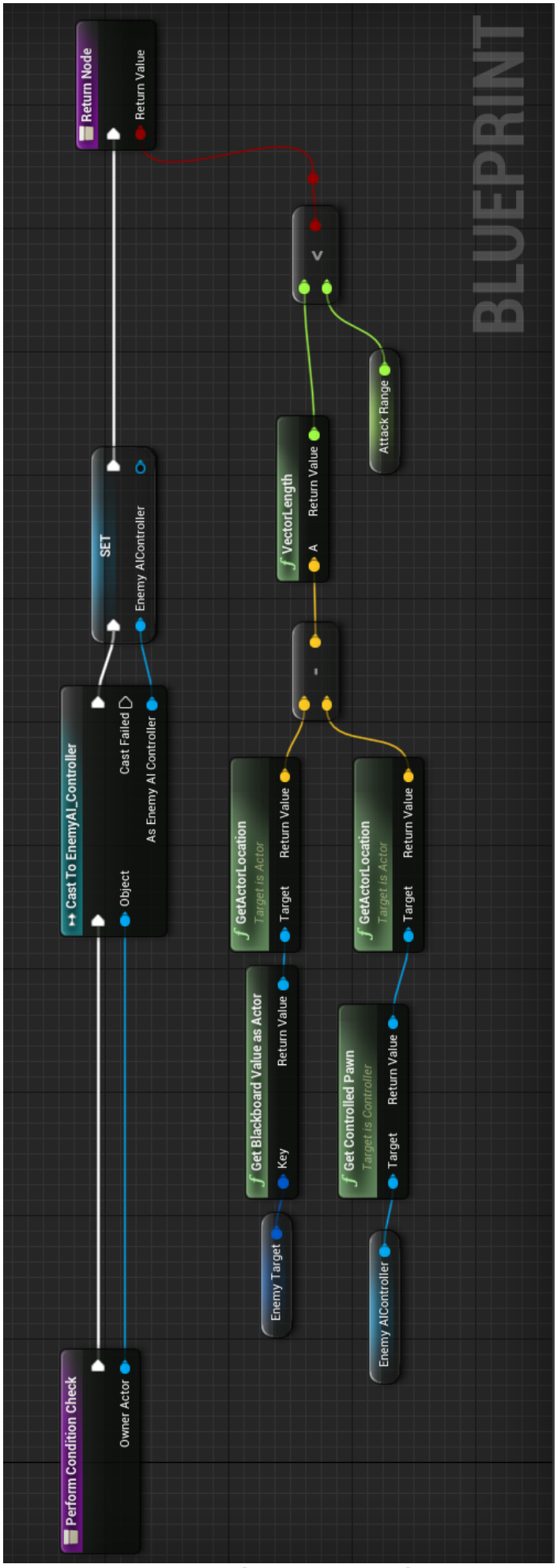


Figure IV-61 Blueprint IsInAttackRange.

- La tâche ‘rotation’ (figure IV-62) permet à l’ennemi de faire face au joueur.

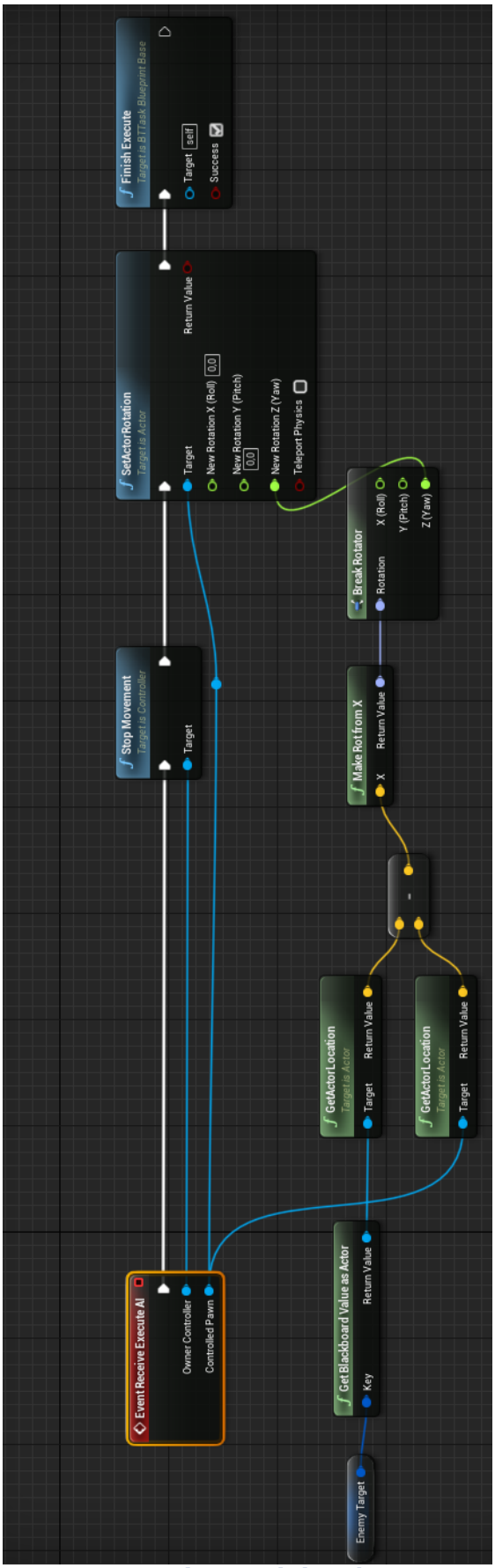


Figure IV-62. Blueprint de la tache rotation.

- La tâche ‘Attack’ (figure IV-63) associée au décorateur ‘*cooldown’* permet à l’ennemi d’attaquer avec son arme principale le joueur. Cette action est répétée périodiquement grâce au décorateur ‘*cooldown’*.

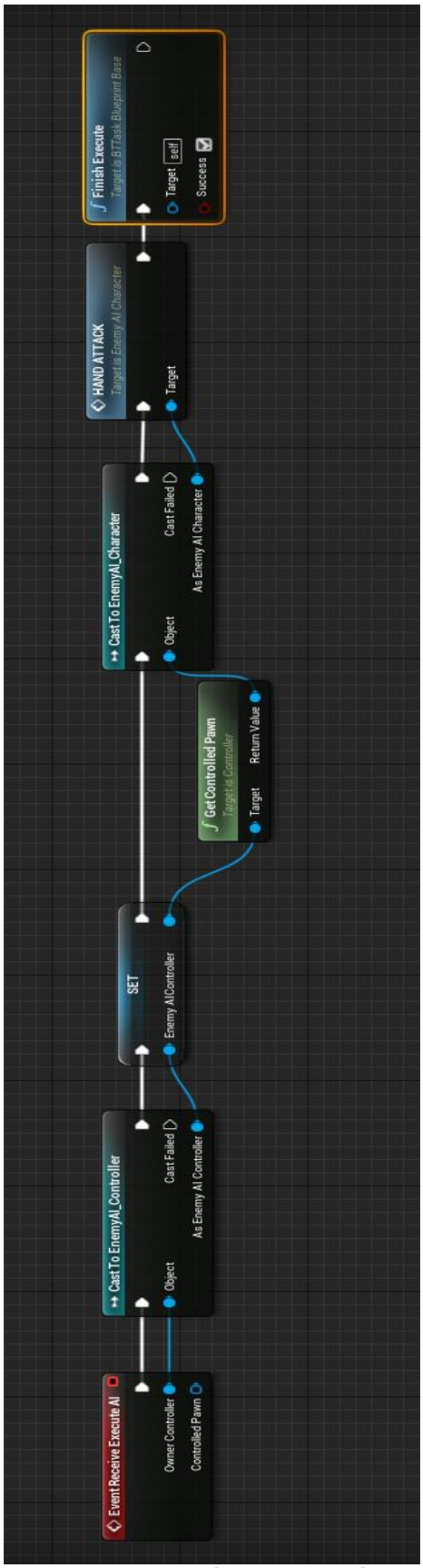


Figure IV-63. Blueprint Attack.

- La tâche ‘BTT_Déplacer_pour_attacker’ (figure IV-64) permet à l’ennemi de se rapprocher du joueur tout en tirant des bombes.

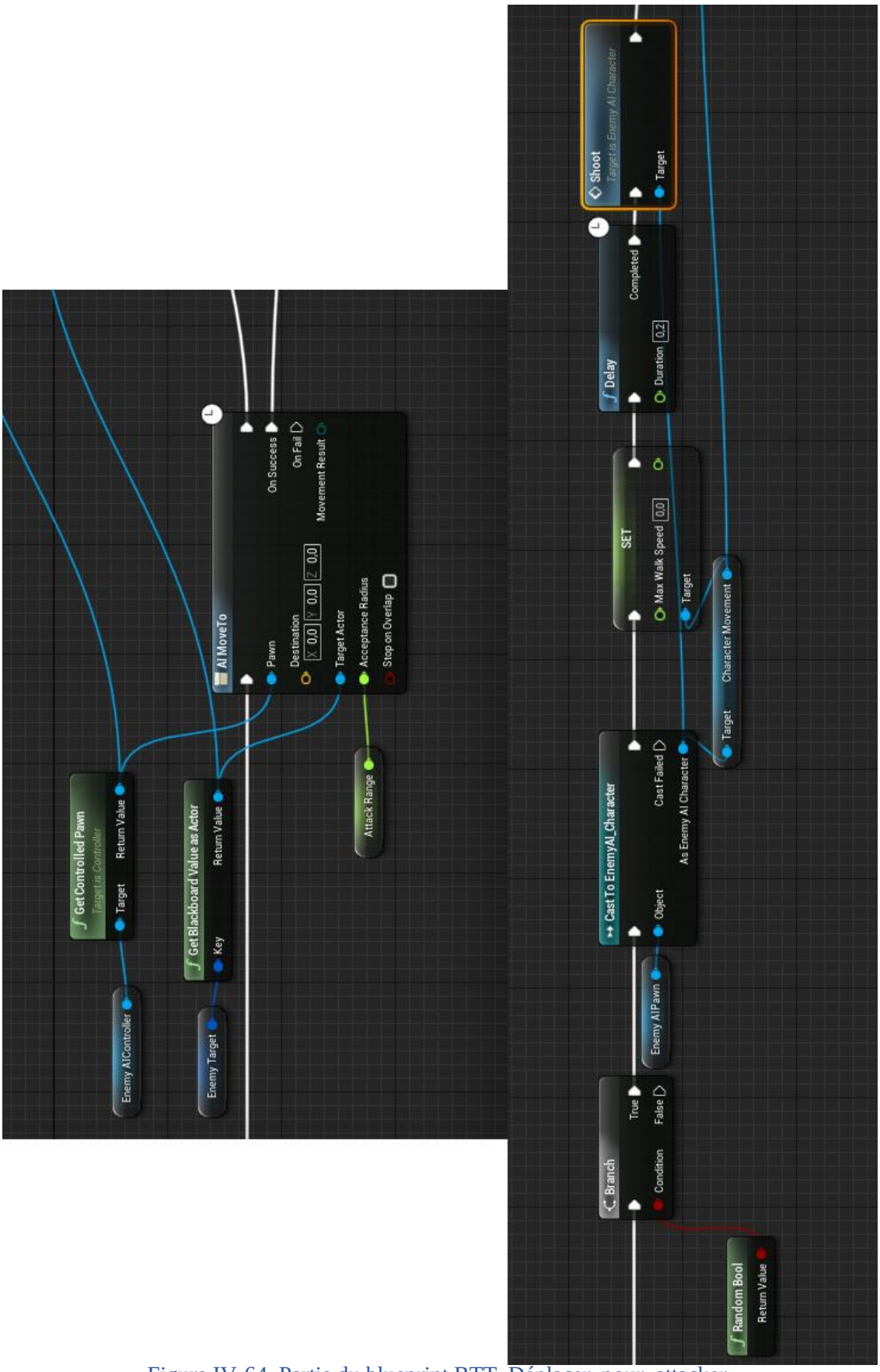


Figure IV-64. Partie du blueprint BTT_Déplacer_pour_attacker.

5. Les interfaces :

Les interfaces jouent un rôle important dans les échanges entre le joueur et l'application. Ce rôle est plus important pour les jeux destinés aux téléphones car la seule interaction possible entre l'utilisateur et l'application est via son écran tactile.

Dans notre cas, afin de pouvoir interagir avec le jeu, un système de reconnaissance du toucher est mis en place grâce à des classes appelées 'Widget'.

Ces widgets sont un ensemble d'éléments 2D comme des images, du texte, des vidéos, des sliders associés à des animations personnalisables et qui sont souvent utilisés comme boutons à clic.

1. **Widget menu principale :**

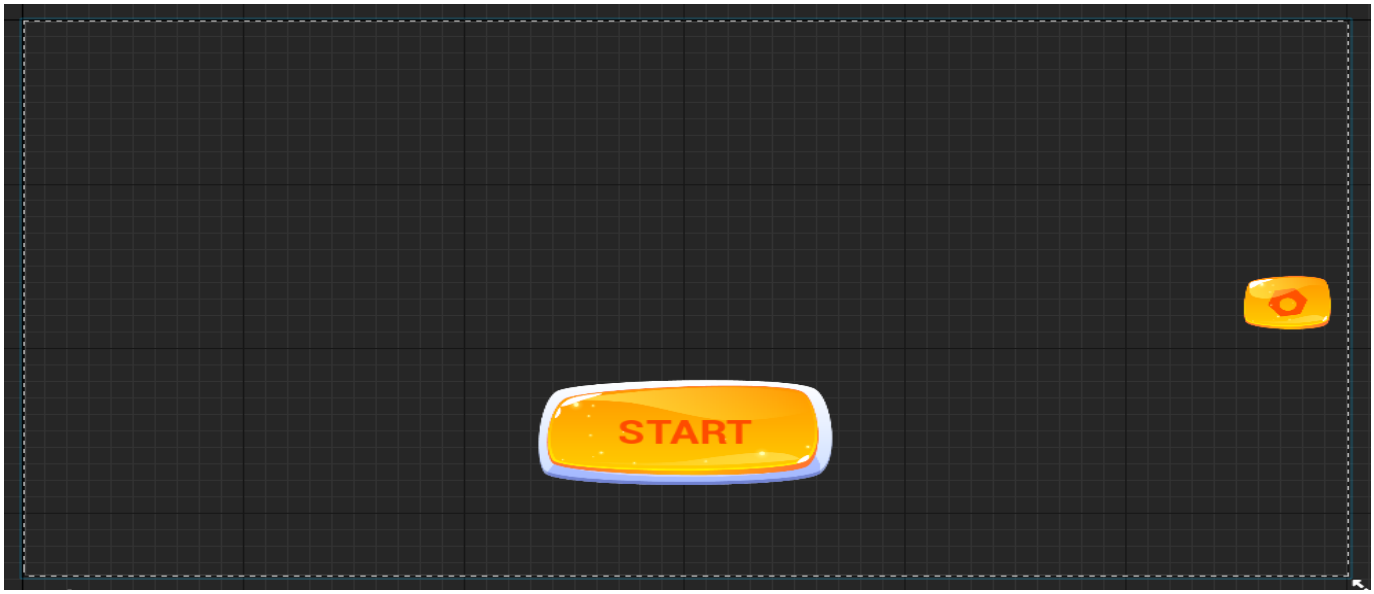


Figure IV-65. Interface du menu principale.

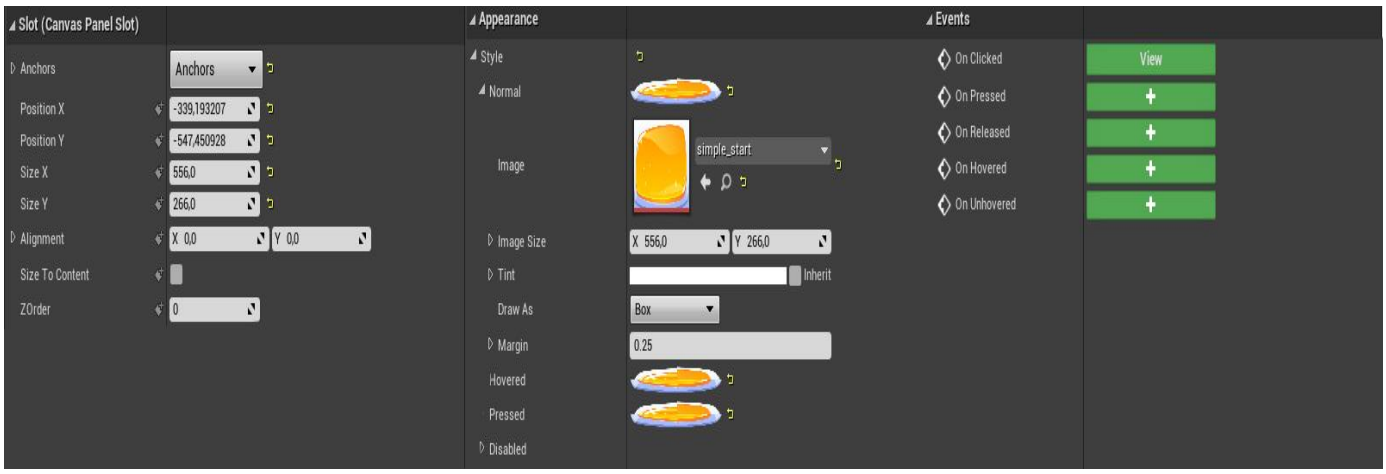


Figure IV-66. Réglage du bouton Start du menu principal.

- A la création du widget, plusieurs animations sont jouées, la bannière de publicité est appelée par la fonction **Event Construct** (figure IV-67) (voir annexe 15).
- **On clicked (start)** et **On clicked (réglage)** (figure IV-68) affichent respectivement les widgets *character* qui est l'interface de personnalisation du personnage et *réglage* qui est l'interface des réglages liés au jeu. Ces interfaces seront affichées sur l'écran à la place de widget *menu principale* (voir annexe 16).

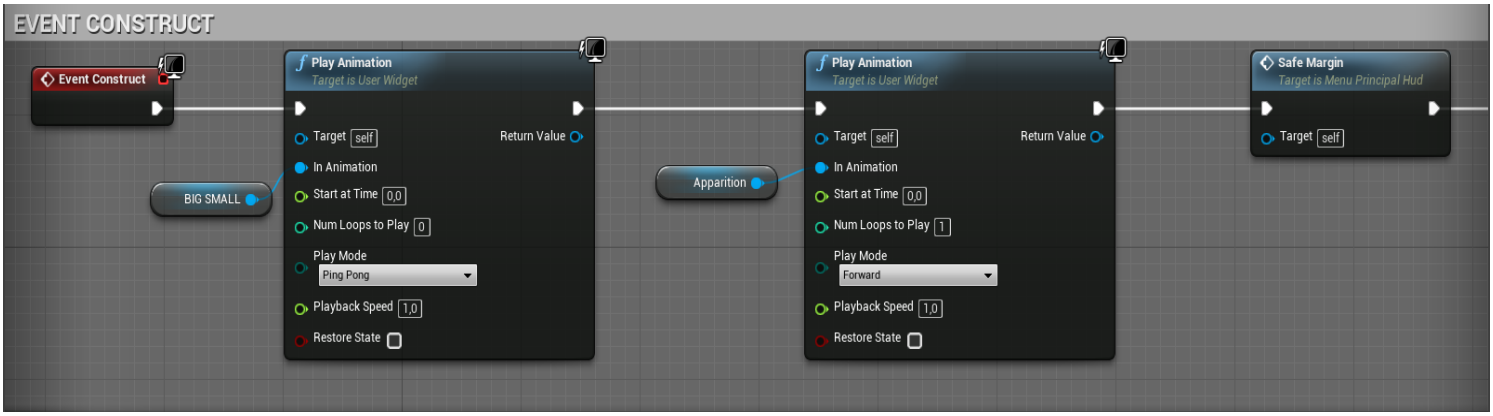


Figure IV-67. Partie du blueprint Event Construct du menu principal.

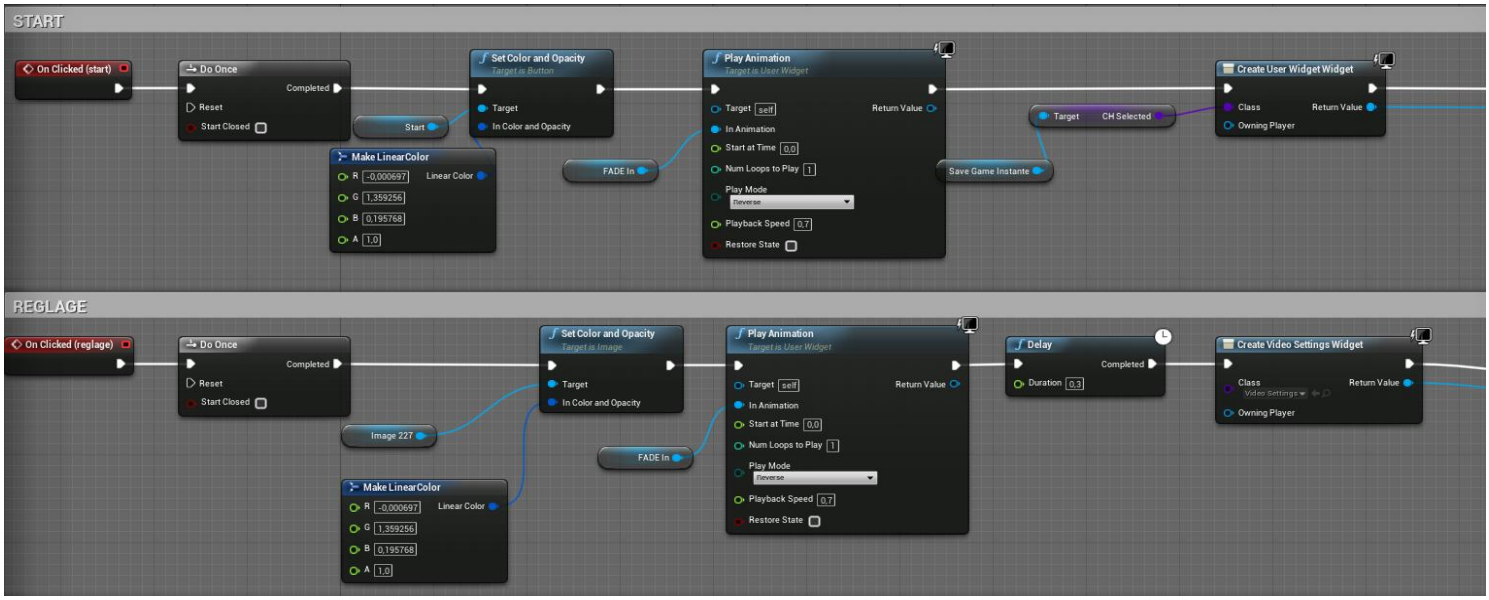


Figure IV-68. Parties des blueprints Start et Réglage du menu principal

2. Widget réglage :

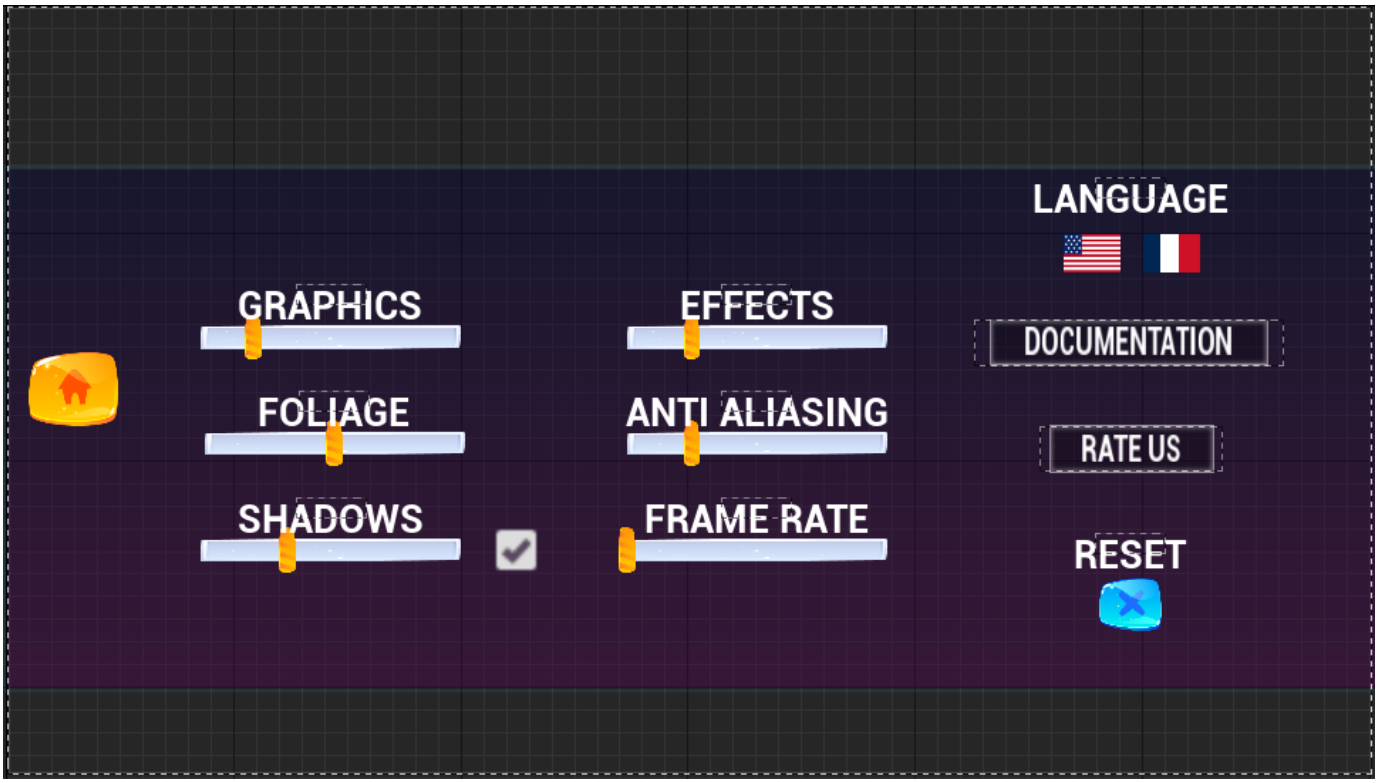


Figure IV-69. Interface du menu réglage.

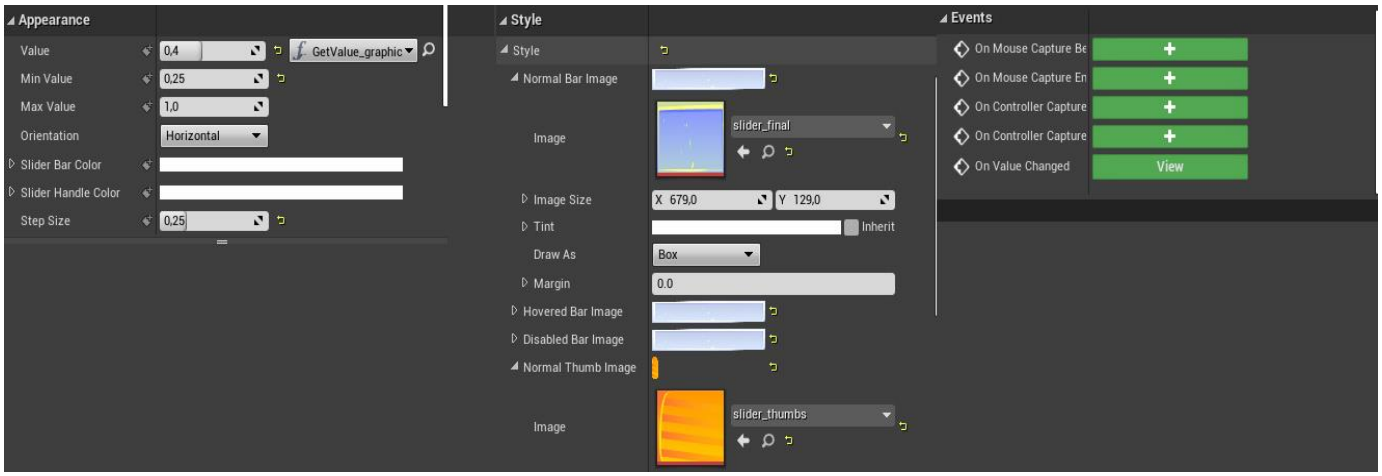


Figure IV-70. Réglage du slider Graphique du menu réglage

- La fonction **Event Construct** (figure IV-71) permet de jouer une animation à la construction du widget et de charger la sauvegarde du jeu.

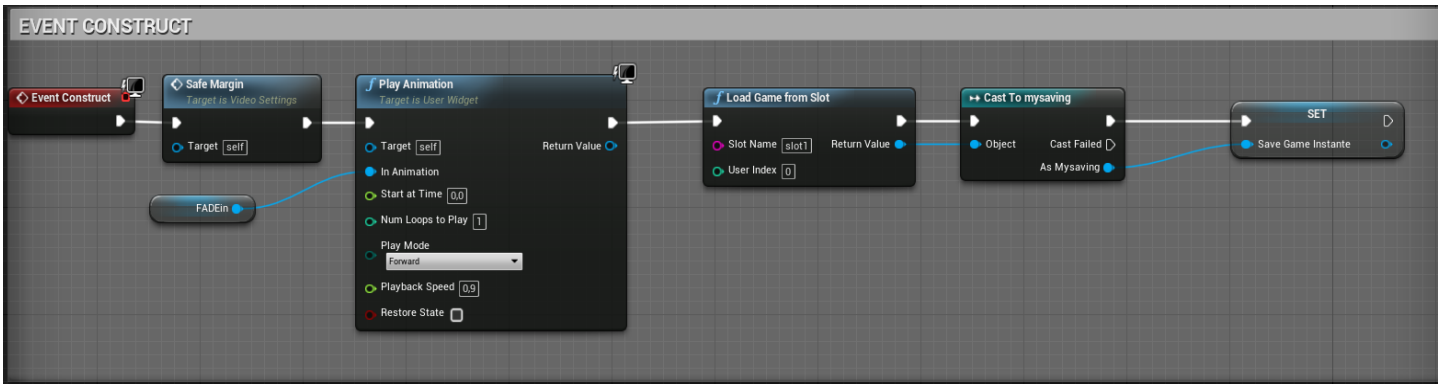


Figure IV-71. Blueprint Event Construct du menu Réglage.

- La fonction **On Clicked(home)** (figure IV-72) permet de créer le widget menu principale et de l'afficher à l'écran (voir annexe 17).

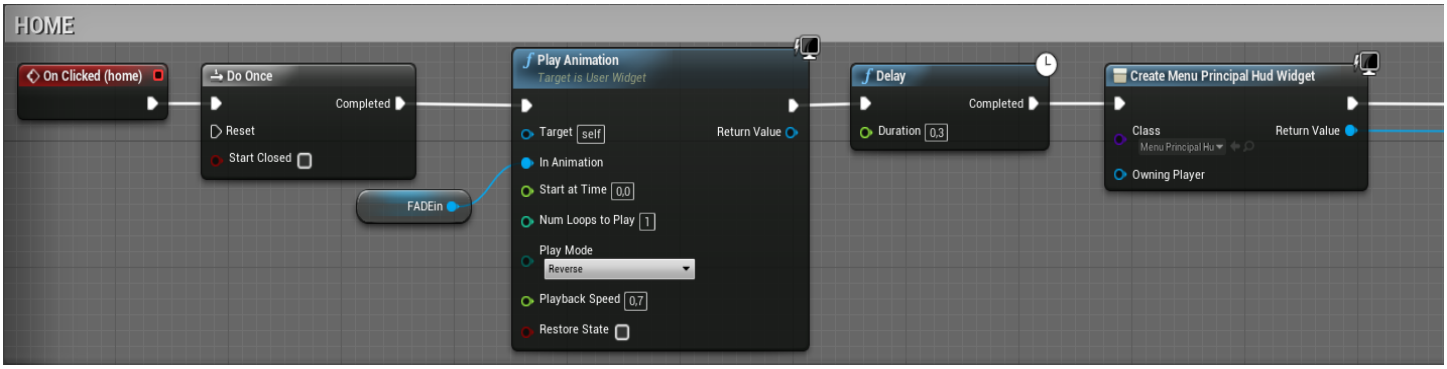


Figure IV-72. Partie du blueprint Home du menu Réglage.

- La fonction **On Value Changed (Slider_Graphic)** (figure IV-73) permet de sauvegarder la nouvelle valeur liée aux réglages des graphisme dans la sauvegarde et d'exécuter une commande pour appliquer le changement de réglage.
- Le même concept est utilisé pour les autres réglages comme le foliage, images par seconde, post process, anti aliasing, et les ombres avec le changement de la commande exécutée (voir annexe 18).

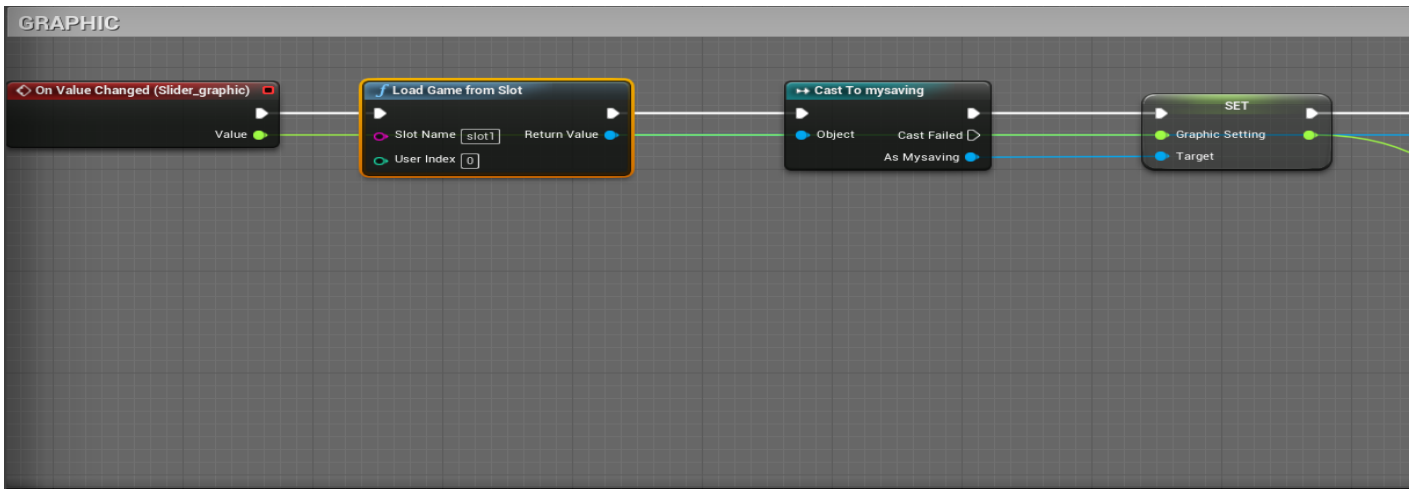


Figure IV-73.Partie du blueprint Graphic du menu Réglage.

- **On Clicked(reset)** (figure IV-74) permet d'effacer les sauvegardes et de remettre les valeurs par défaut.
- **On Clicked (EN)** et **On Clicked (FR)** (figure IV-74) permettent de choisir la langue du jeu.
- **On Clicked(rate_us)** et **On Clicked(documentation)** (figure IV-74) permettent de lancer des liens web.

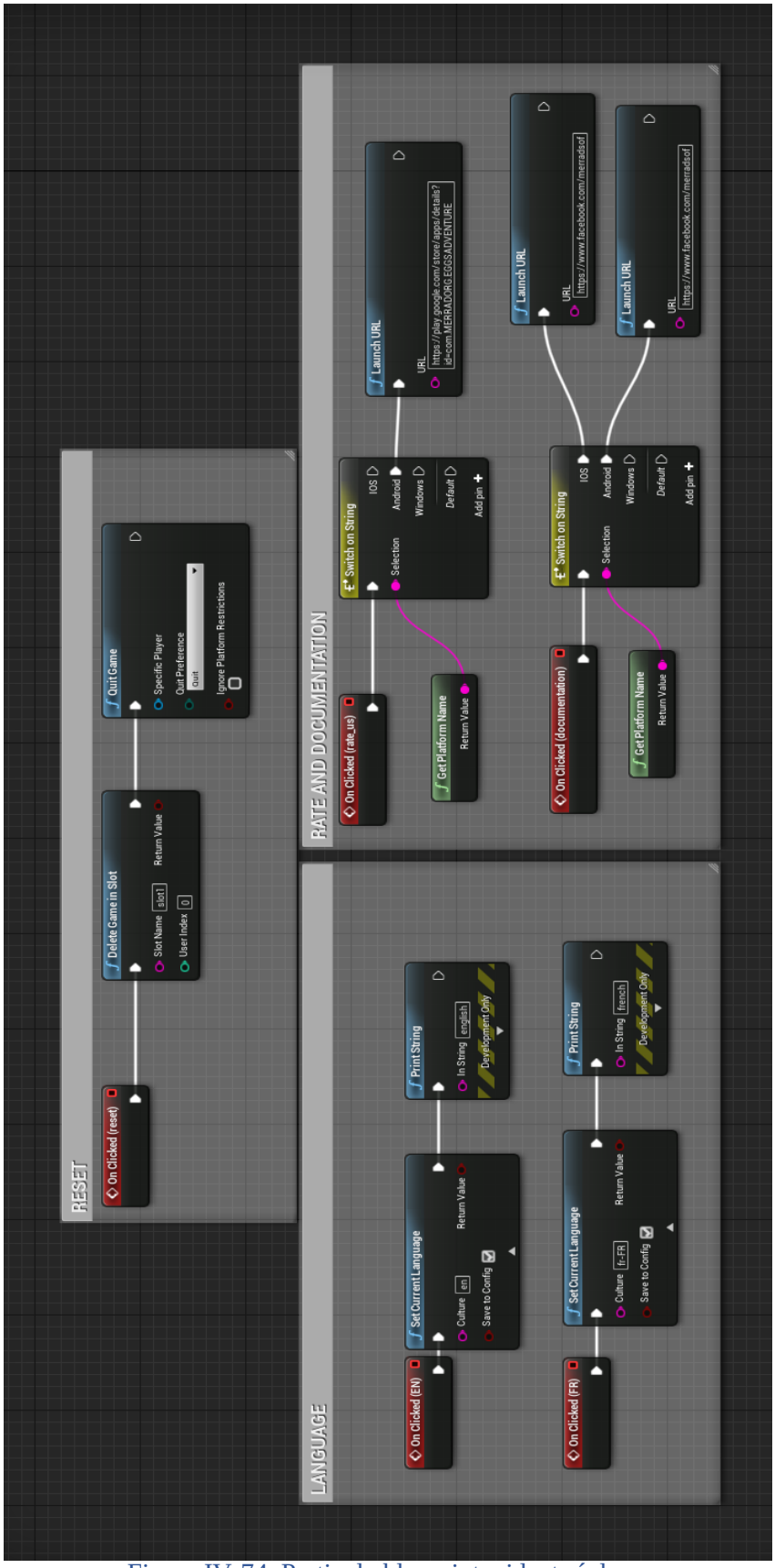


Figure IV-74. Partie du blueprint widget réglage.

3. Widget menu back

Le menu back est un ensemble de boutons toujours disponible en haut de l'écran, il permet un accès rapide au bouton son, au bouton quitter et d'afficher le nombre de pièces.

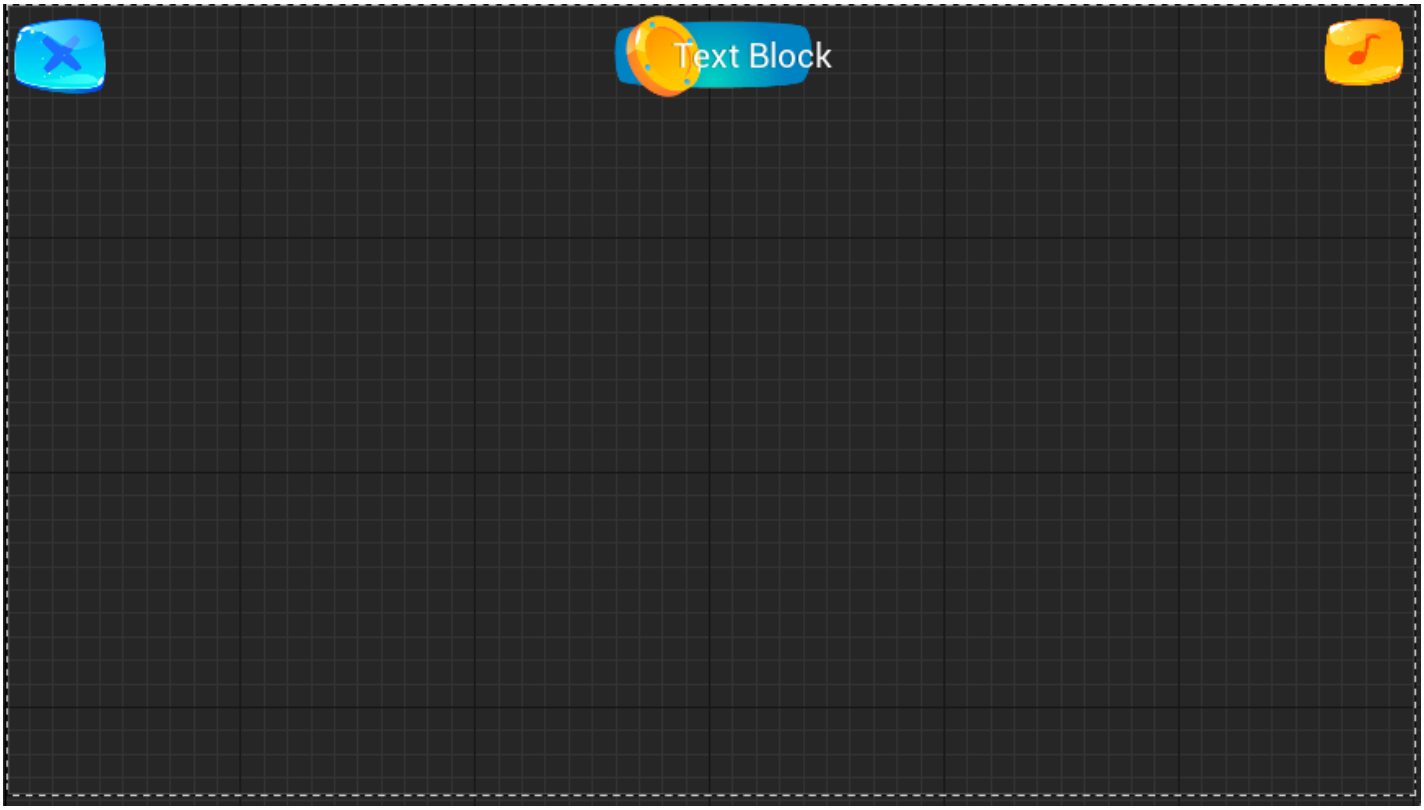


Figure IV-75. Interface du menu back.

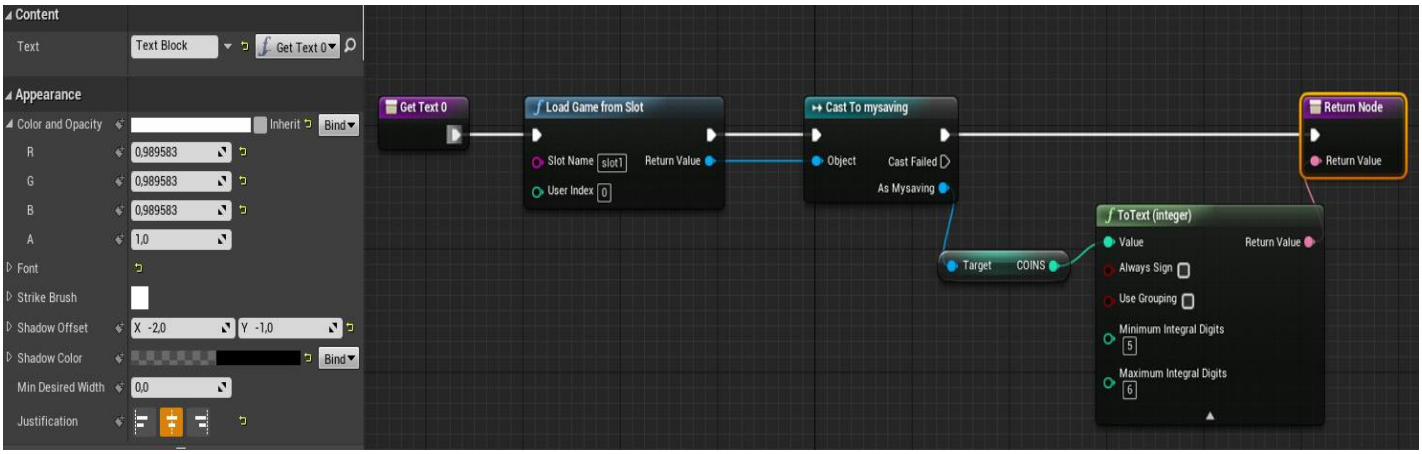


Figure IV-76. Réglage du texte nombre de pièce.

- La fonction **Event Construct** comme précédemment vu lance des animations et récupère des sauvegardes (voir annexe 19).
- La fonction **On Clicked (Exit)** permet de quitter le jeu.
- Les fonctions **On Clicked (son_off)** et **On Clicked (son on)** permettent de désactiver ou d'activer le son et d'afficher le bouton correspondant à l'action inverse dans le widget (voir annexe 20).

4. Widget caractère

Le menu caractère comporte un menu de customisation du personnage et de son amélioration.

Il permet d'augmenter les points de vie, des points de dégât de l'arme contre des pièces, Il permet aussi d'acheter des nouvelles tenues pour le personnage (voir annexe 21).

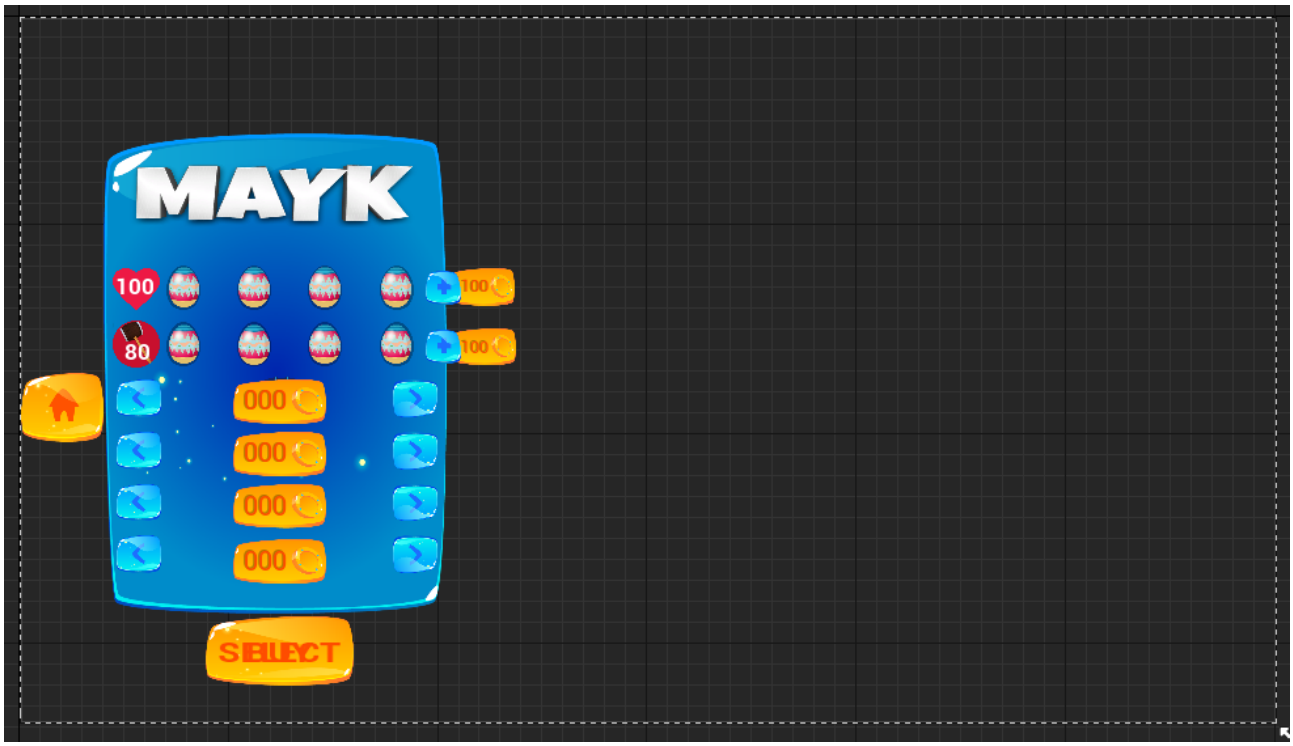


Figure IV-77. Interface du widget caractère.

- Le bouton home permet de créer le widget menu principale et de l'afficher
- Le bouton acheter permet d'acheter le personnage.
- Le bouton select permet de sélectionner le personnage et de passer à la sélection de niveau.
- Les boutons suivant et précédent de la sélection de tenue incrémentent et décrémentent respectivement un entier qui sert à retourner à chaque fois une tenue différente qui sont stockées dans un vecteur d'objets.
- L'utilisateur peut visualiser une tenue sans pour autant l'acheter, et en quittant le menu de sélection de caractère, c'est la dernière tenue achetée que l'utilisateur a visualisé qui sera sélectionnée.

L'idée générale des achats dans ce menu se résume à ce qu'on :

- Vérifie si le joueur dispose d'assez de pièces pour modifier son personnage.
- À soustraire le nombre de pièces nécessaires à l'achat.
- Améliorer le personnage ou lui changer de tenue (figure IV-78 et IV-79).
- De sauvegarder le nombre de pièces restantes ainsi que d'augmenter les points de vie ou de dégâts du personnage et de faire passer un article de customisation de non acheté à acheté.
- D'afficher un widget No Money si le joueur ne dispose pas d'assez de pièce.

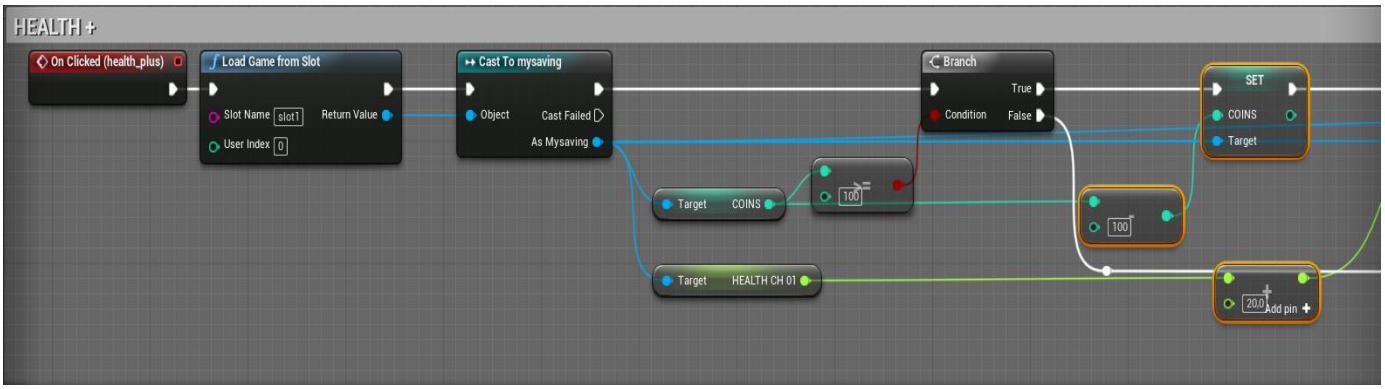


Figure IV-78. Partie du blueprint pour Augmenter les points de vie.

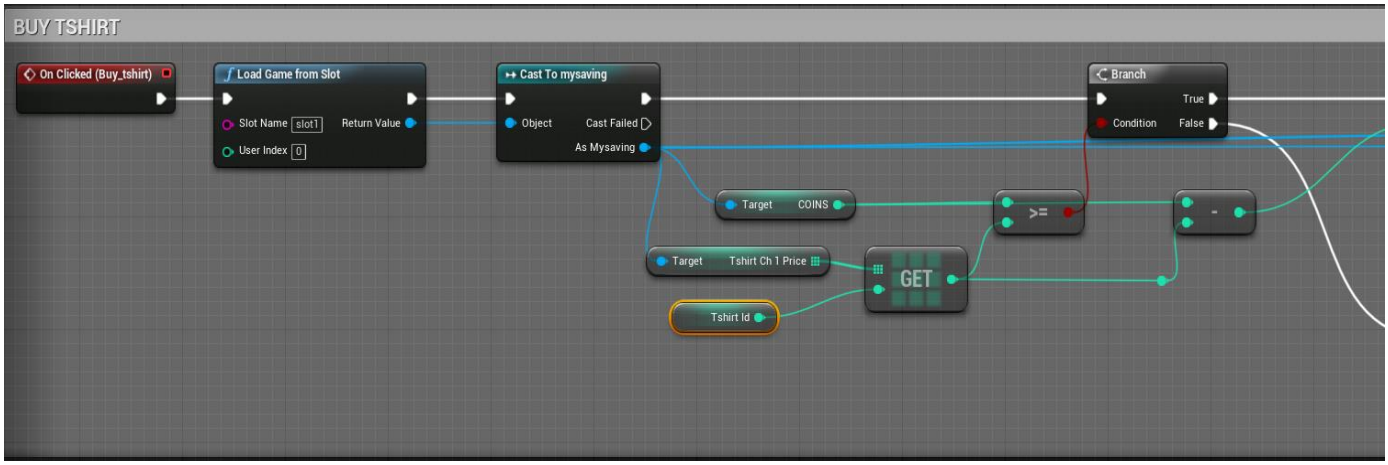


Figure IV-79. Partie du blueprint Buy Tshirt.

5. Widget Level

Les interfaces Level permettent d'afficher les détails de chaque niveau ainsi qu'une barre de progression, elles permettent aussi de lancer une partie dans le niveau choisi.

Figure IV-80. Interface du widget Level2.

- Le bouton home permet de créer le widget menu principale et de l'afficher.
- Le bouton suivant permet de passer à l'interface niveau suivante (figure IV-81).
- Le bouton Go permet de lancer la partie (figure IV-81).

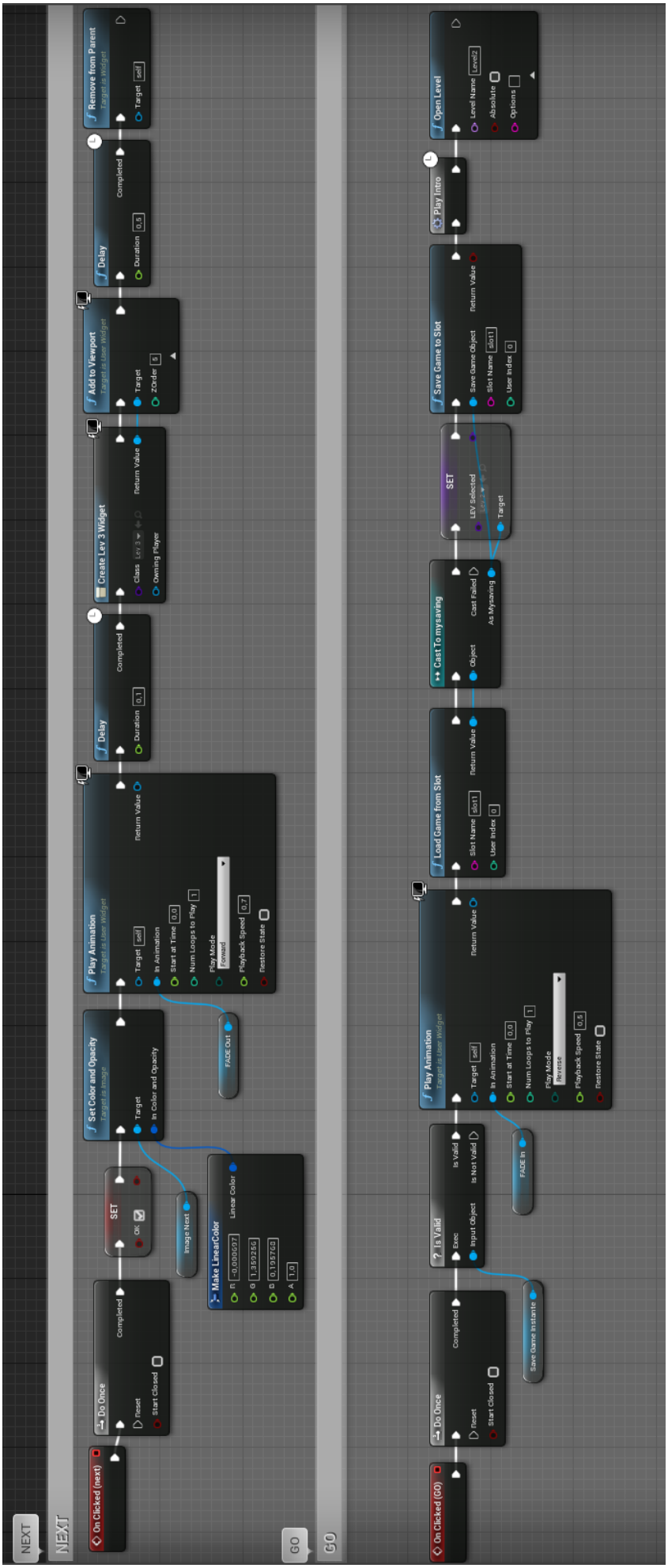


Figure IV-81. Parties du widget blueprint Level.

6. Widget No Money

Le menu No money s'affiche quand le joueur essaye d'acheter un article ou d'améliorer son personnage sans avoir le nombre de pièces nécessaires.

L'interface affiche un message avertissant que le joueur ne dispose pas d'assez de pièces.

L'interface offre la possibilité au joueur de regarder des vidéos de publicité pour gagner des pièces (figure IV-83).

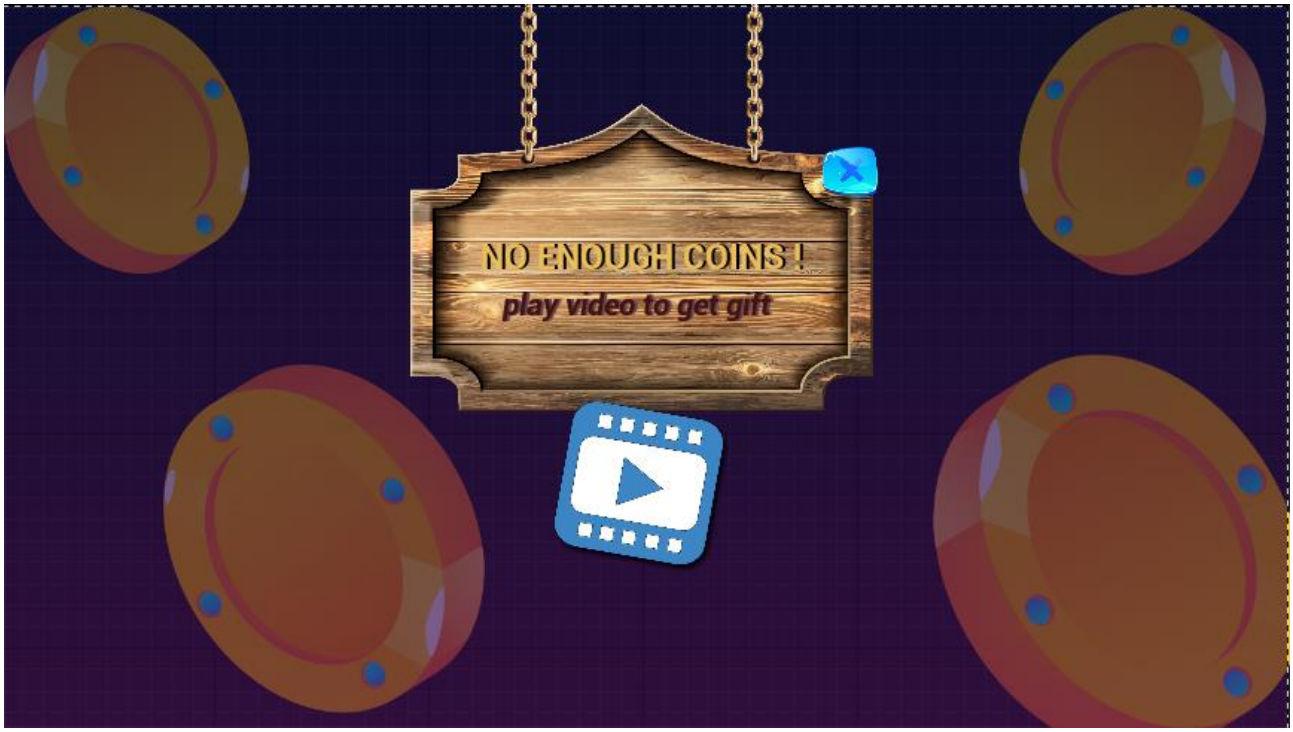


Figure IV-82. Interface du widget No money.

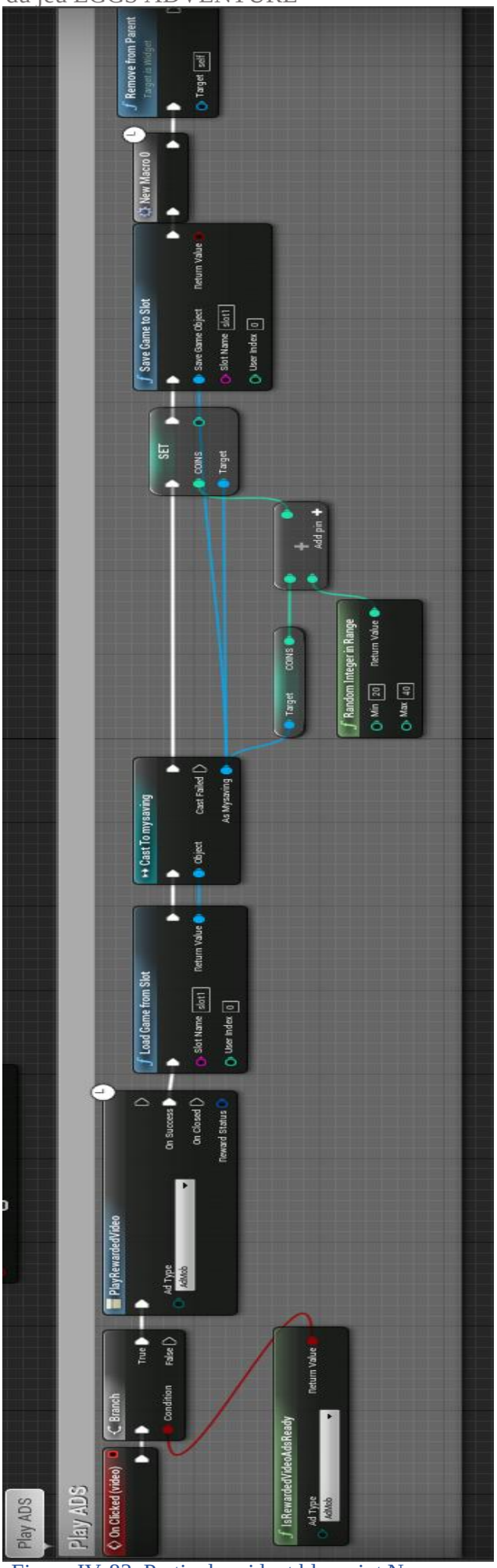


Figure IV-83. Partie du widget blueprint No money.

7. Widget HUD Mobile

HUD Mobile l'un des widgets les plus important, il permet au joueur d'interagir avec son caractère pendant une partie et d'afficher les informations nécessaires au bon déroulement du jeu comme le nombre de pièces gagnées, le niveau de santé, le nombre de vies et de temps restants.

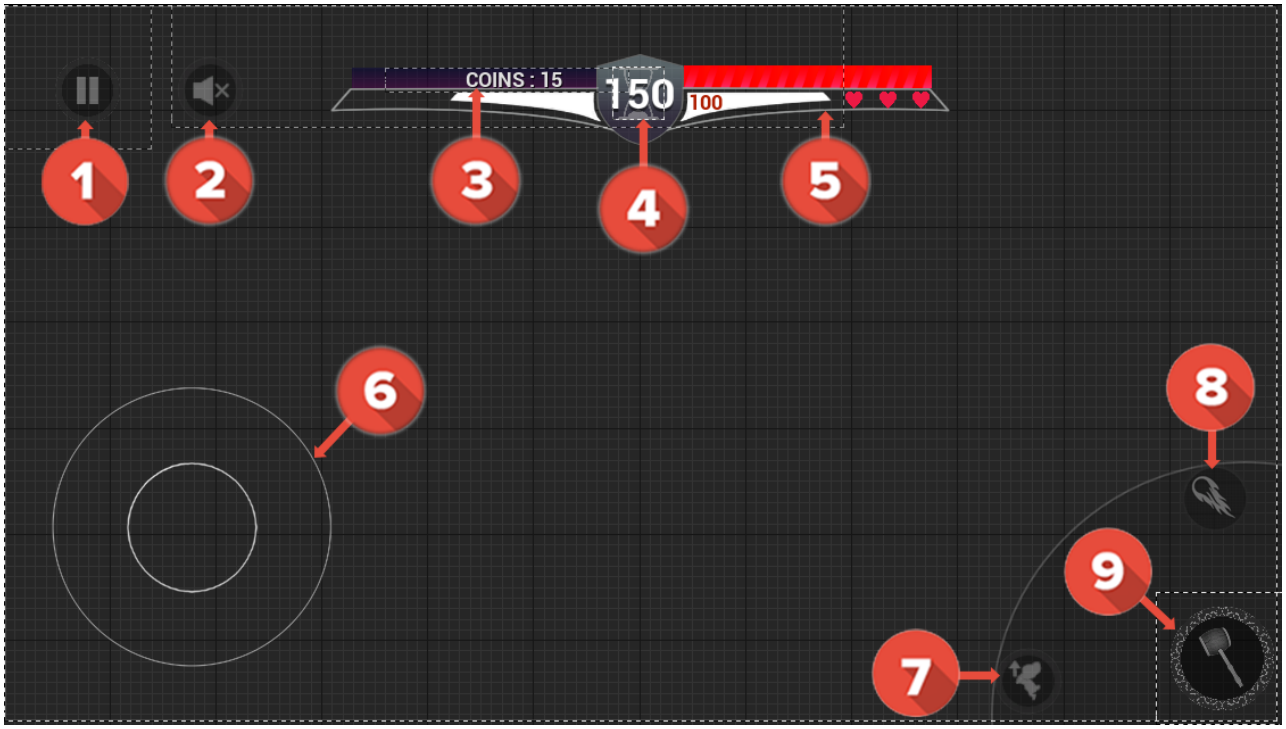


Figure IV-84. Interface du widget HUD mobile.

Les fonctionnalités de ce widget sont :

1. Mise en pause du jeu.
2. Activation ou désactivation du son.
3. Affichage du nombre de pièces gagnées et de victimes dans la partie.
4. Affichage du temps restant.
5. Affichage du niveau de vie et du nombre de vies restantes.
6. Affichage d'un joystick pour faire déplacer le caractère.
7. Affichage d'un bouton pour sauter.
8. Affichage des boutons pour utiliser des bombes.
9. Affichage d'un bouton pour utiliser l'arme principale.

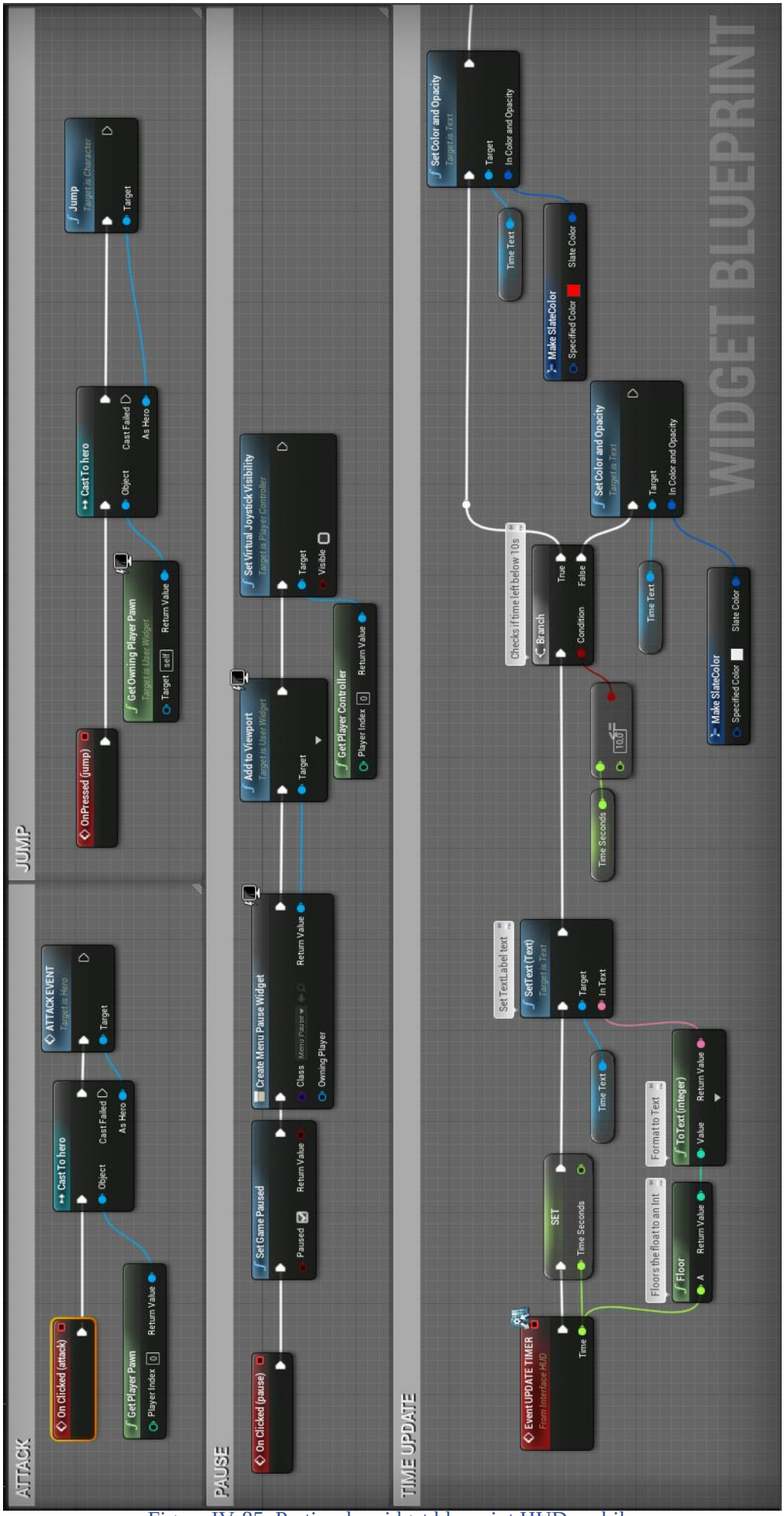


Figure IV-85. Parties du widget blueprint HUD mobile.

- Les fonctions **attack** et **jump** (figure IV-85) appellent les fonctions définies déjà dans le caractère blueprint pour effectuer respectivement la tâche d'attaque avec l'arme principale et de sauter.
- La fonction **On Clicked(pause)** (figure IV-85) permet de mettre en pause le jeu et de créer un widget pause qui contiendra des informations pour joueur.
- La fonction **Event Update Timer** (figure IV-85) permet de mettre à jour chaque seconde le minuteur et de l'afficher sur l'écran.

10. Widget pause et Game over

Les widget pause et Game over permettent respectivement de mettre en pause et d'arrêter la partie. Le menu pause est affiché par l'utilisateur tandis que le menu Game over est affiché automatiquement à la fin du temps ou l'épuisement des vies.

Les deux menus affichent des informations comme le nombre de pièces gagnées et de victimes, en plus de cela le menu pause affiche le temps restant et le niveau de vie ainsi que le nombre de vies restantes.

Le menu pause permet d'accéder à un menu réglage, de quitter la partie pour le menu principal, de retourner à la partie et de quitter complétement le jeu.

Le menu Game over permet de retourner au menu principal, de rejouer la même partie, ou de quitter complétement le jeu.

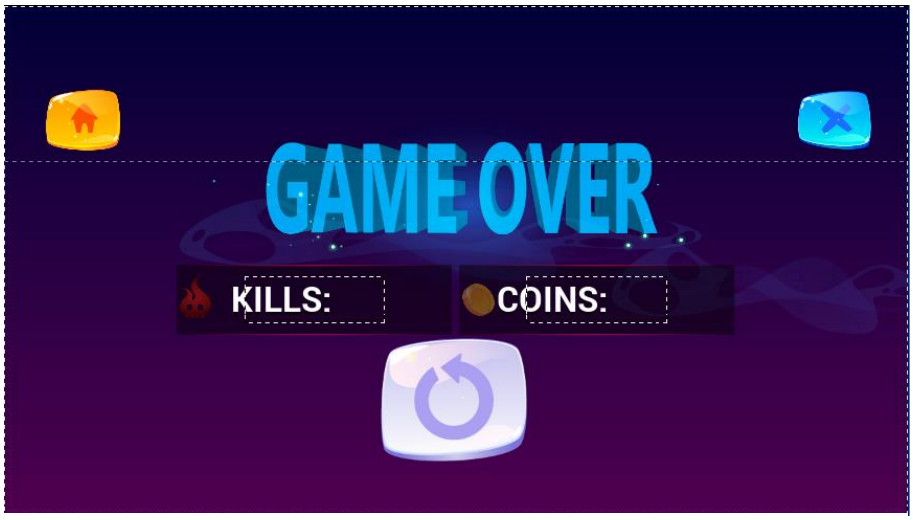


Figure IV-86. Interface du widget menu Game over.

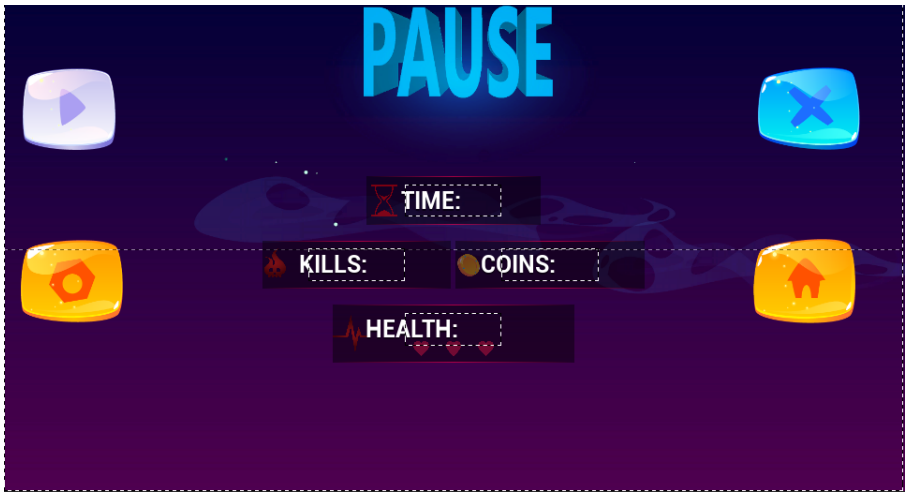


Figure IV-87. Interface du widget menu pause.

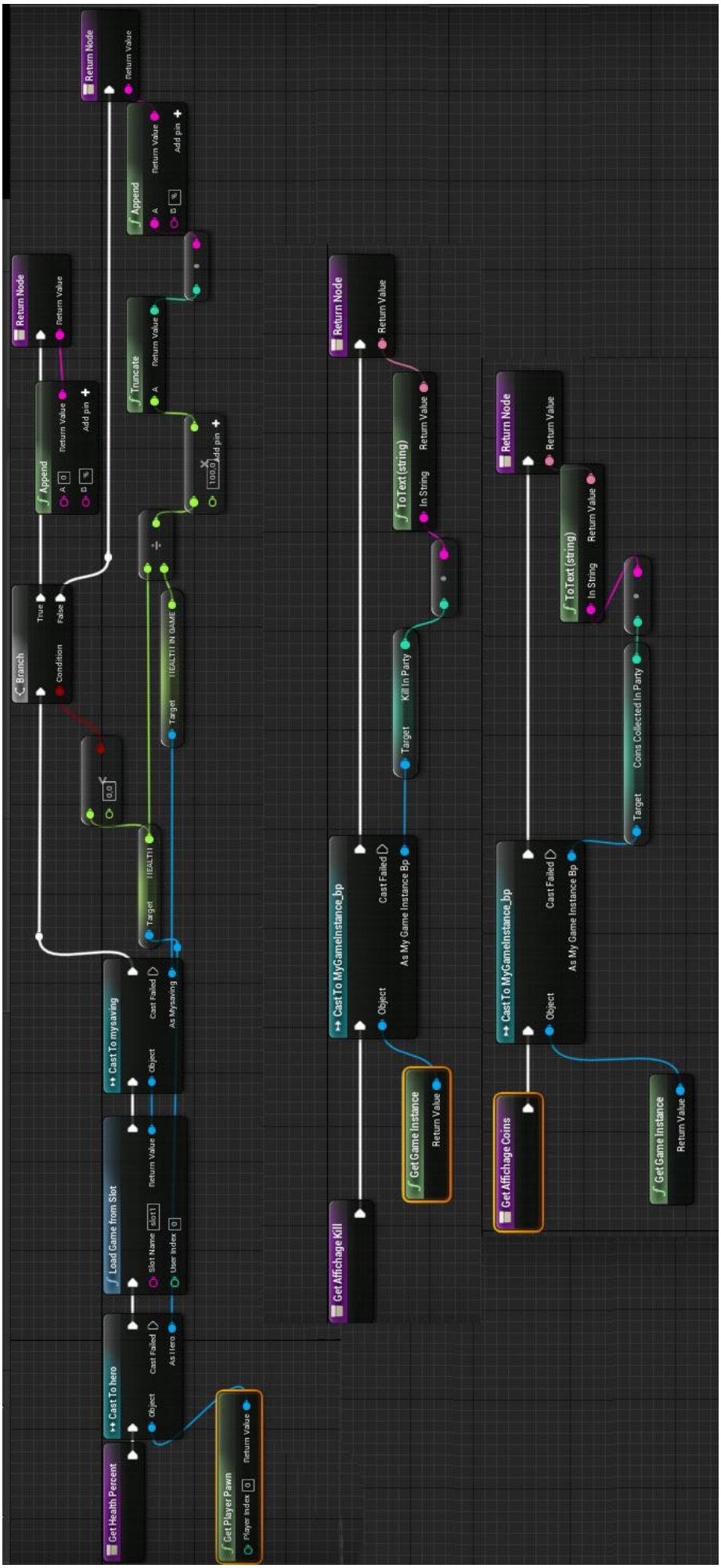


Figure IV-88. Parties du widget blueprint stats menu pause et Game over.

Chapitre V : Les publicités et publication du jeu EGGS ADVENTURE

1. Publicité et rémunération

Les applications et les jeux qui sont publiés sur le store de Google et Apple suivent souvent trois modes de rémunérations différents :

- Revenus par publicités : les applications utilisant des publicités sont souvent gratuites, le créateur rajoute des blocks de publicités dans différents emplacements dans l'application qui s'afficheront au cours de l'exécution de cette dernière.
- Revenus par vente de l'application : les applications payantes sont souvent utilisées dans le cas où les revenus par publicités ne sont pas assez rentables pour le créateur, ou dans le cas d'un jeu avec grand budget.
- Revenus par micro transaction : de plus en plus, les applications qui sont disponibles sur les stores intègrent des systèmes d'achat dans le jeu. Un utilisateur peut ainsi dépenser de l'argent pour acquérir de nouvelles fonctionnalités ou des améliorations dans le jeu

Un jeu peut combiner plusieurs méthodes de rémunérations suivant le Game Play du jeu, le marché visé et le nombre d'utilisateurs.

1.1 Publicité et le système Google AdMob dans EGGS ADVENTURE.

Le Jeu EGGS ADVENTURE est destiné principalement à un public jeune, de ce fait une rémunération par achat et micro transaction n'est pas efficace vu que les jeunes personne ne dispose pas de carte de payement et de moyen de payement en ligne. C'est pour cela que le moyen le plus judicieux pour rentabiliser son application reste via des publicités et dans ce cas Google AdMob a été utilisé.

Google AdMob

AdMob est une société de publicité sur mobile fondée par Omar Hamoui. Elle a été créée en 2006. Google l'a rachetée en novembre 2009 pour 750 millions de dollars [15].

AdMob offre la possibilité d'intégrer des publicités dans des applications sur mobile tournant sur différentes plateformes comme Android et IOS.

Figure V-1 Logo de Google AdMob.

Les étapes pour intégrer des publicités dans un jeu et comme exemple EGGS ADVENTURE :

1. Accéder au site de Google AdMob '*https://admob.google.com*'.

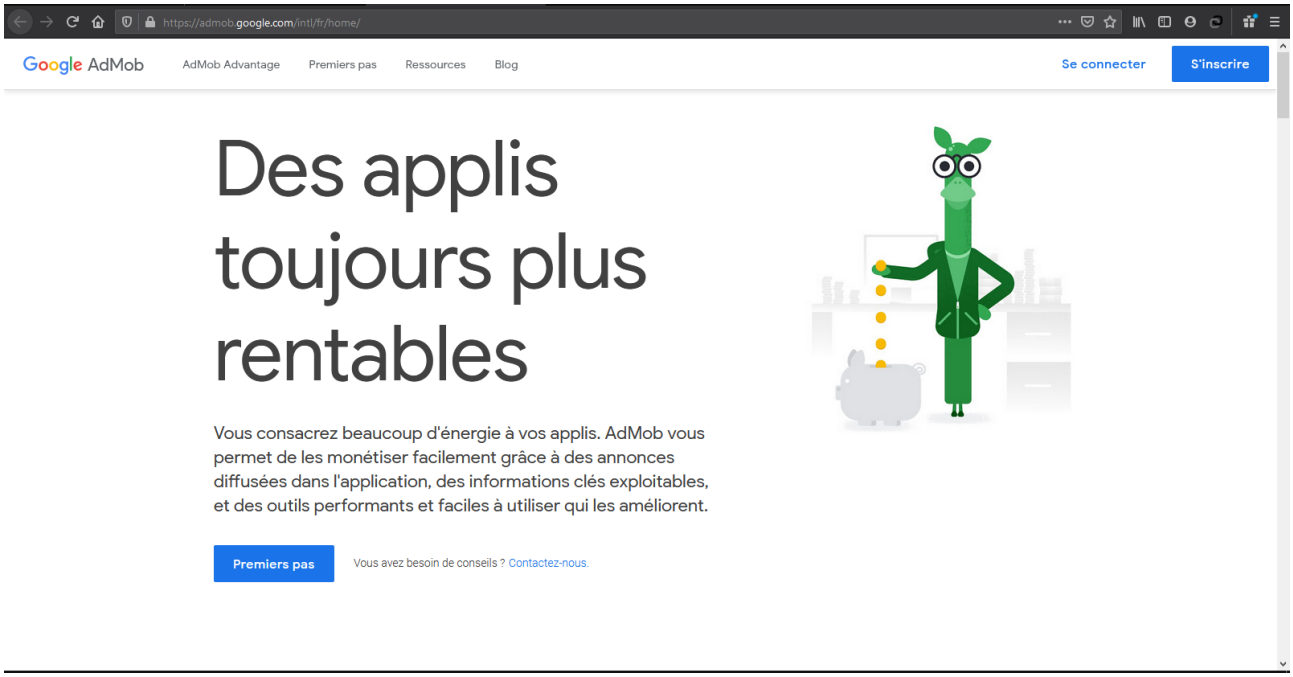


Figure V-2. Page de création d'un compte Google AdMob.

2. Créer un compte gratuitement sur Gmail et se connecter à ce dernier.
3. Connecter son compte Gmail au compte AdMob.

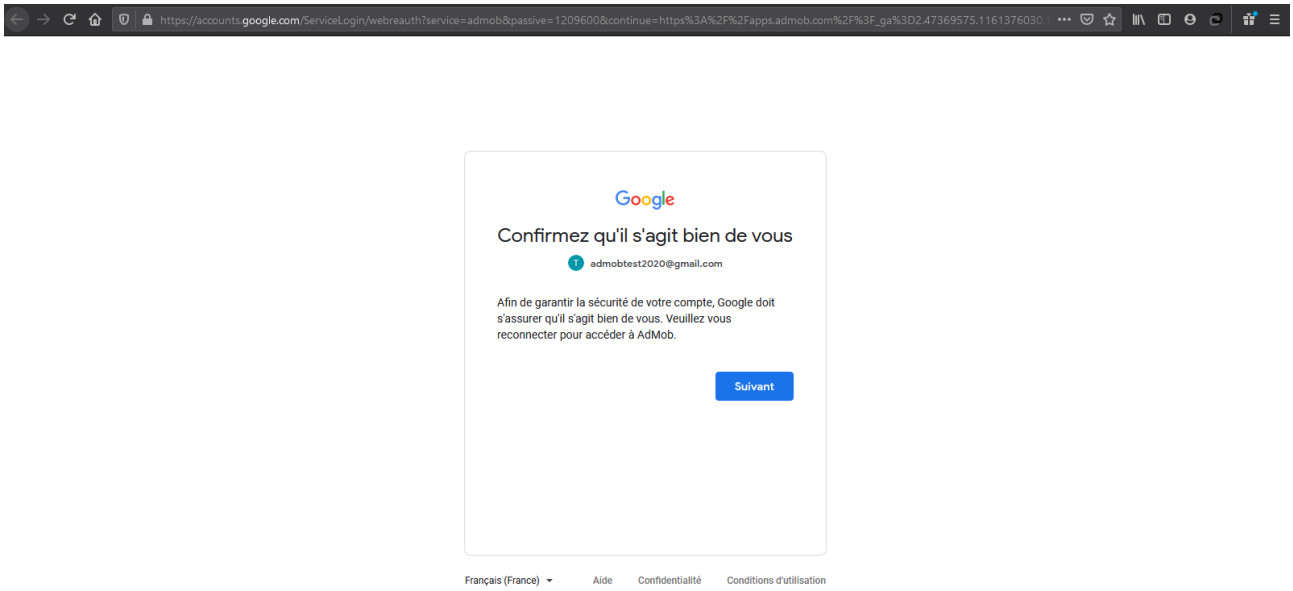


Figure V-3.Page de connexion au compte AdMob avec un compte Gmail.

4. Remplir les formulaires d'inscription.

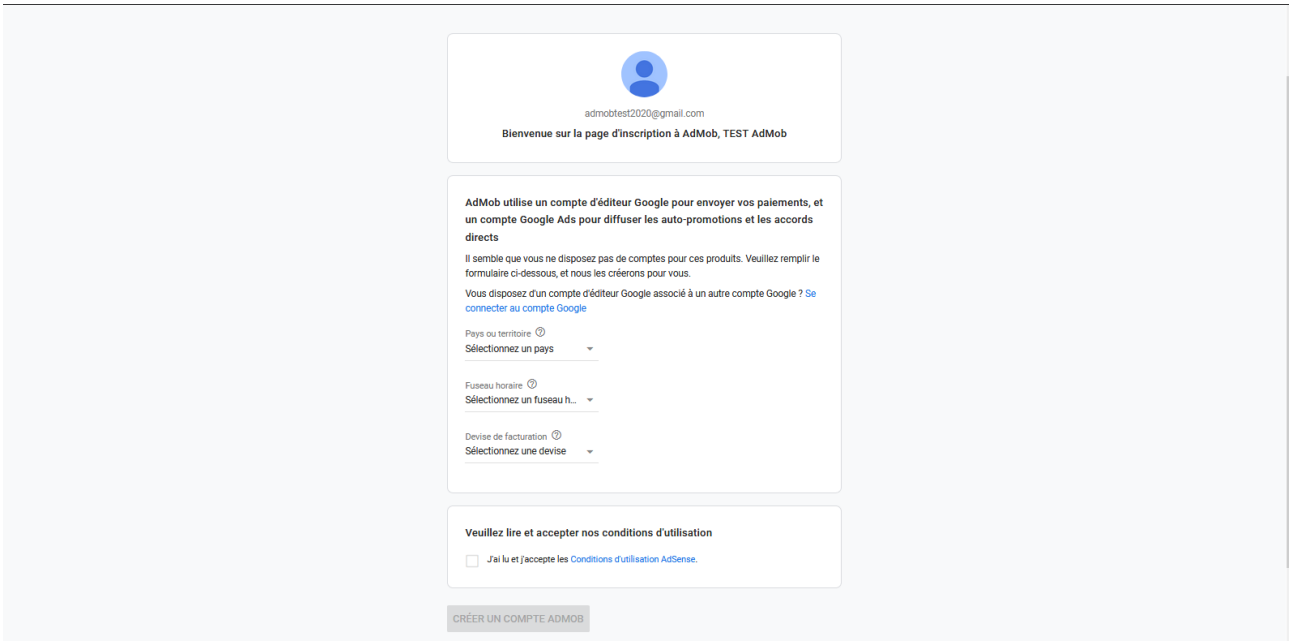


Figure V-4. Formulaire d'inscription.

A la fin de l'inscription, la page d'accueil sera affichée.

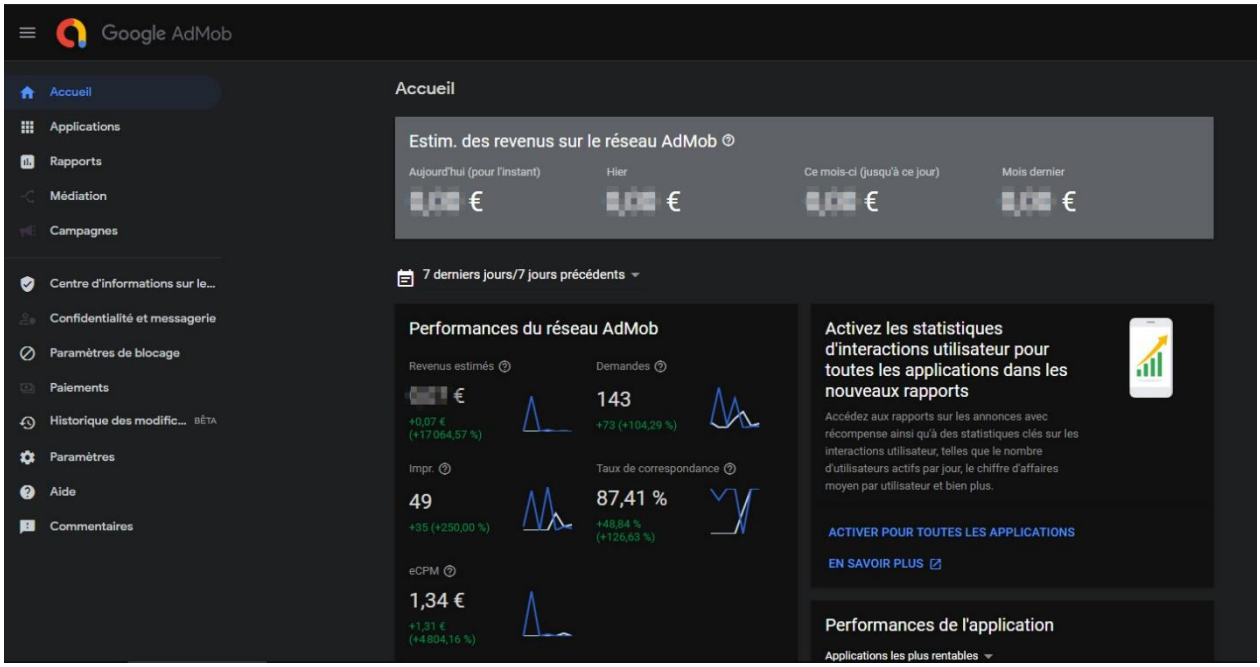


Figure V-5. Page d'accueil du site Google AdMob.

5. Sélectionner Application dans le menu à gauche, puis ajouter une nouvelle application et suivre les étapes qui suivent.

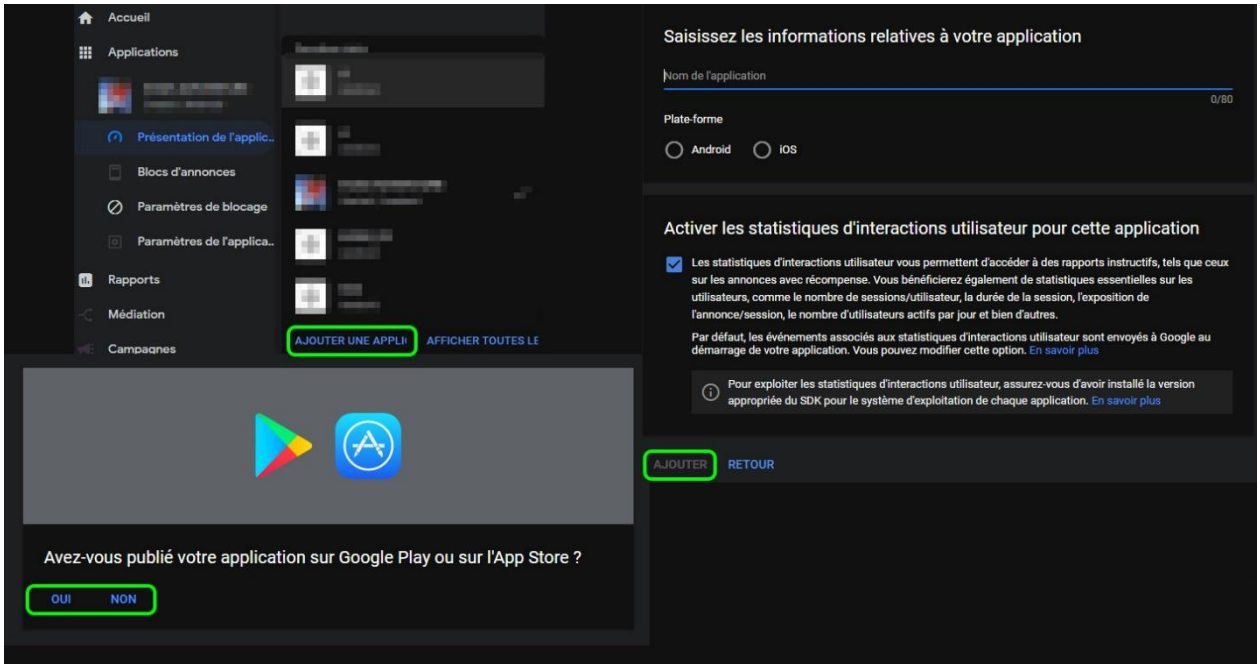


Figure V-6. Etape de création d'une application dans Google AdMob.

6. Sélectionner l'application dans le menu Application et choisir le sous menu présentation de l'application pour voir les informations de l'application.

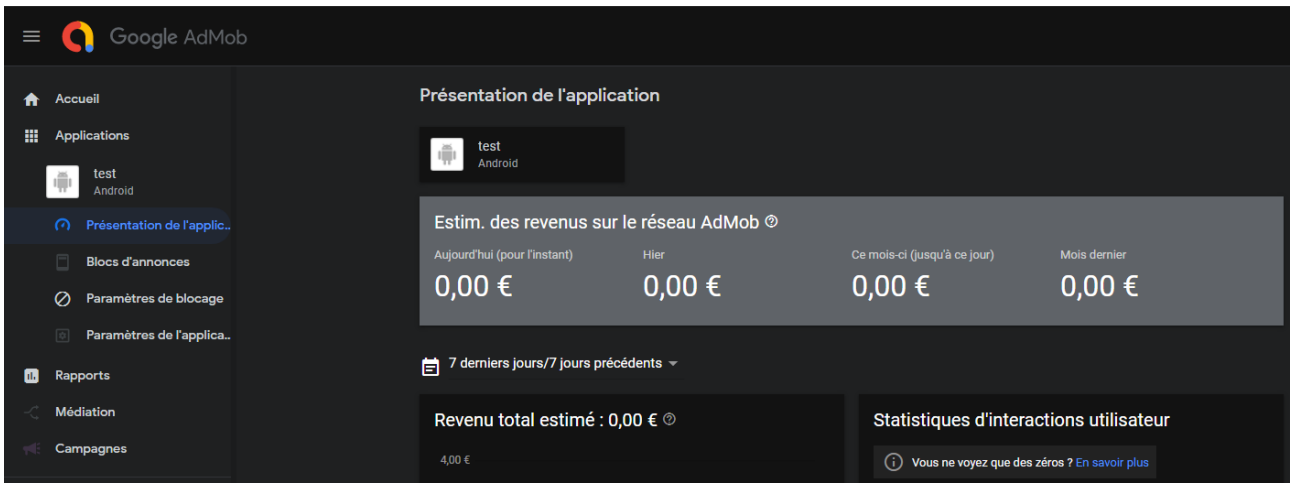


Figure V-7. Exemple d'information d'une application sur Google AdMob.

7. Sélectionner Bloc d'annonces dans le menu Application et suivre les étapes.

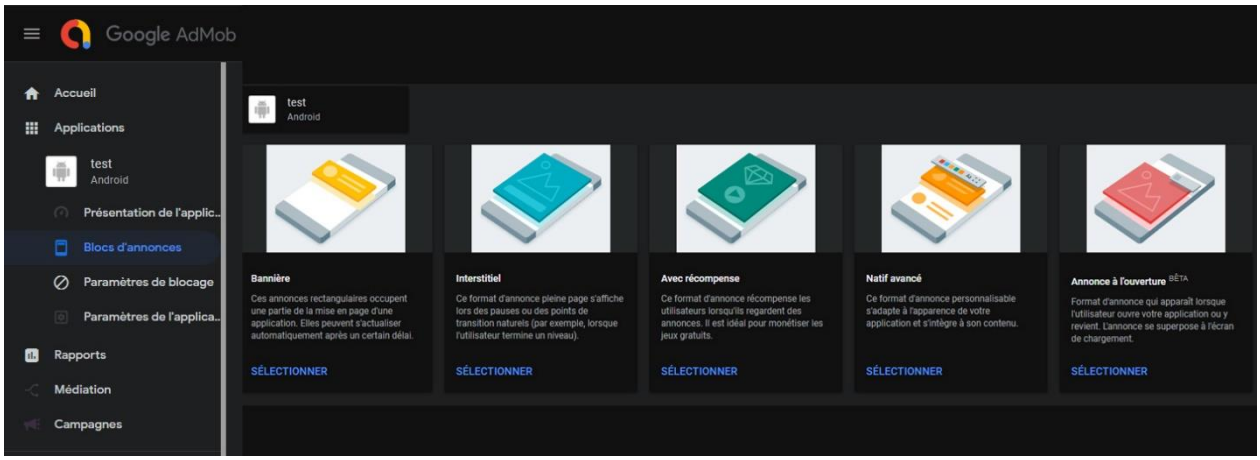


Figure V-8. Etape de création d'un bloc d'annonce.

- Les blocs d'annonces sont organisés sous 5 groupes, et dans EGGS ADVENTURE trois d'entre eux sont utilisés :
  - Bannière : Ce sont des publicités qui s'affichent en bas de l'image en continue.
  - Interstitiel : Ce sont des publicités qui s'affichent en plein écran lors des pauses ou de transitions de menus.
  - Avec récompense : Ce sont généralement des vidéos que l'utilisateur regarde pour gagner des bonus différents et dans le cas d'EGGS ADVENTURE ce sont des pièces que le joueur gagne.

8. Saisir le nom du bloc d'annonces

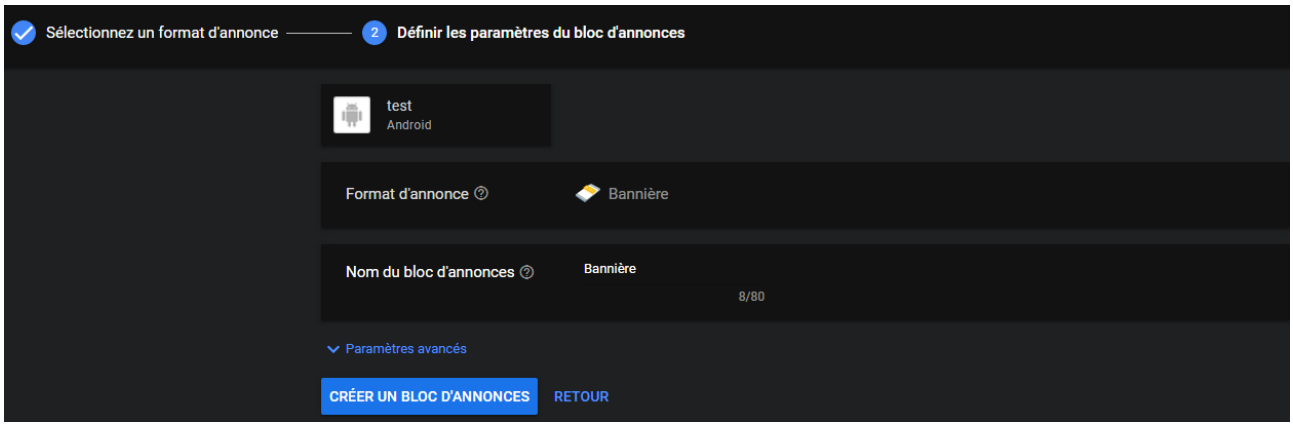


Figure V-9. Page de création d'un bloc d'annonces.

9. Lors de la création d'un bloc d'annonces, un ID d'application et un ID de block d'annonces sont générés et seront utilisés ultérieurement dans l'application.

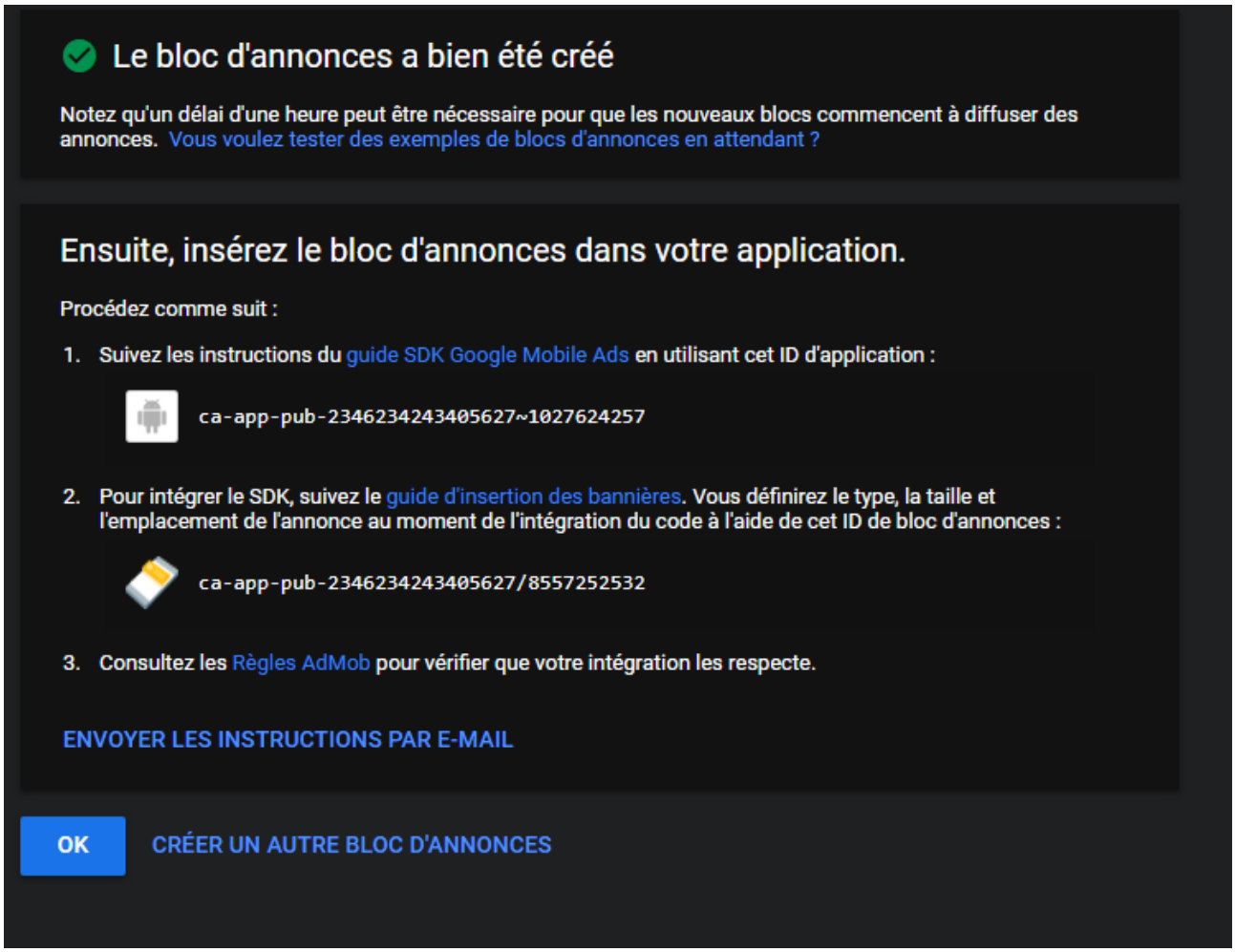


Figure V-10. Exemple d'ID d'application et d'ID de bloc d'annonces.

10. A la fin de la création des blocs d'annonces, les différents ID seront copiés dans le moteur Unreal pour pouvoir être utilisés.

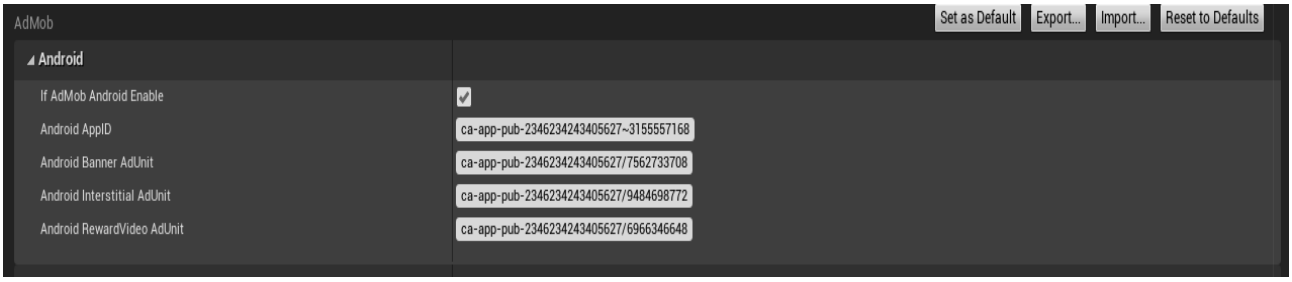


Figure V-11. Intégration des blocs d'annonces dans Unreal.

11. Le développeur doit faire appel aux différents blocs d'annonces via les widgets Blueprint.

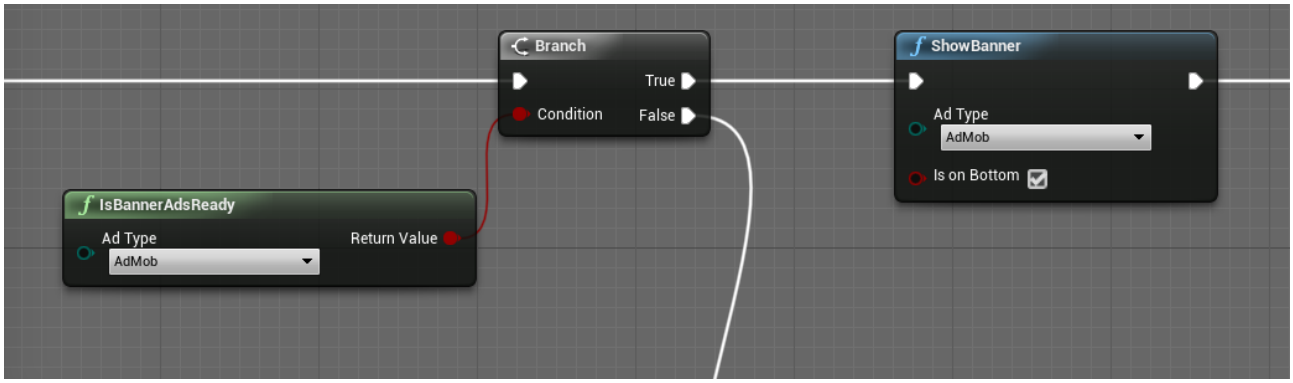


Figure V-12. Appel à la fonction d'affichage du bloc d'annonces du type bannière.

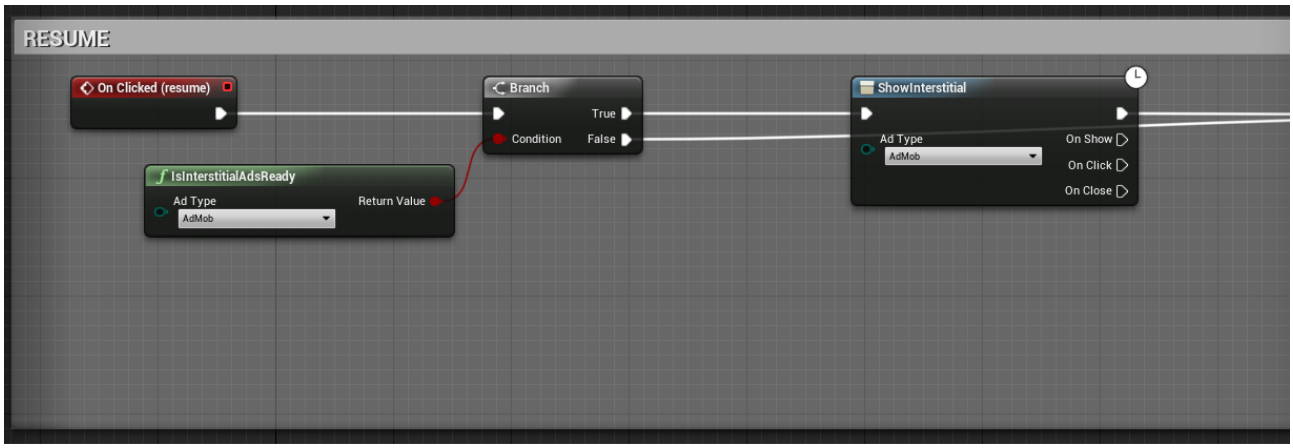


Figure V-13. Appel à la fonction d'affichage du bloc d'annonces du type interstitiel.

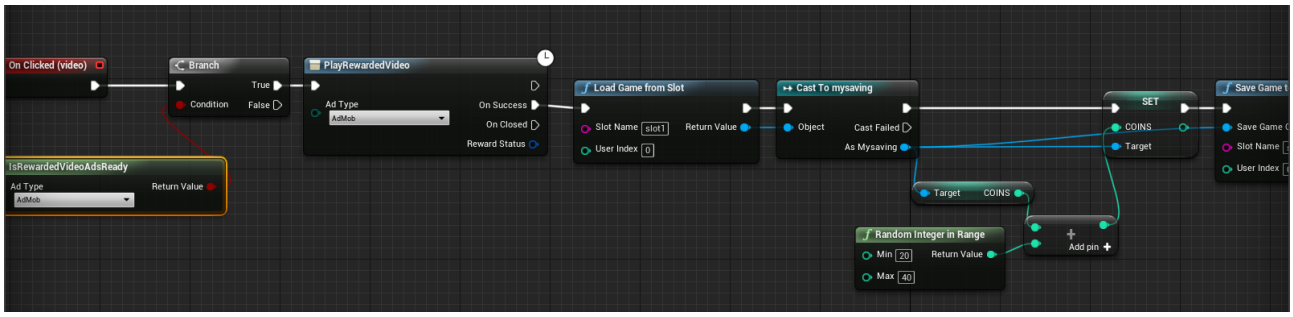


Figure V-14. Appel à la fonction d'affichage du bloc d'annonces du type avec récompense.

- Si une publicité de type avec récompense s'est correctement finie, **On success** (figure V-14) sera exécutée et le joueur recevra des pièces en bonus, dans le cas contraire ou le joueur a arrêté la publicité avant sa fin, aucun changement ne sera effectué.

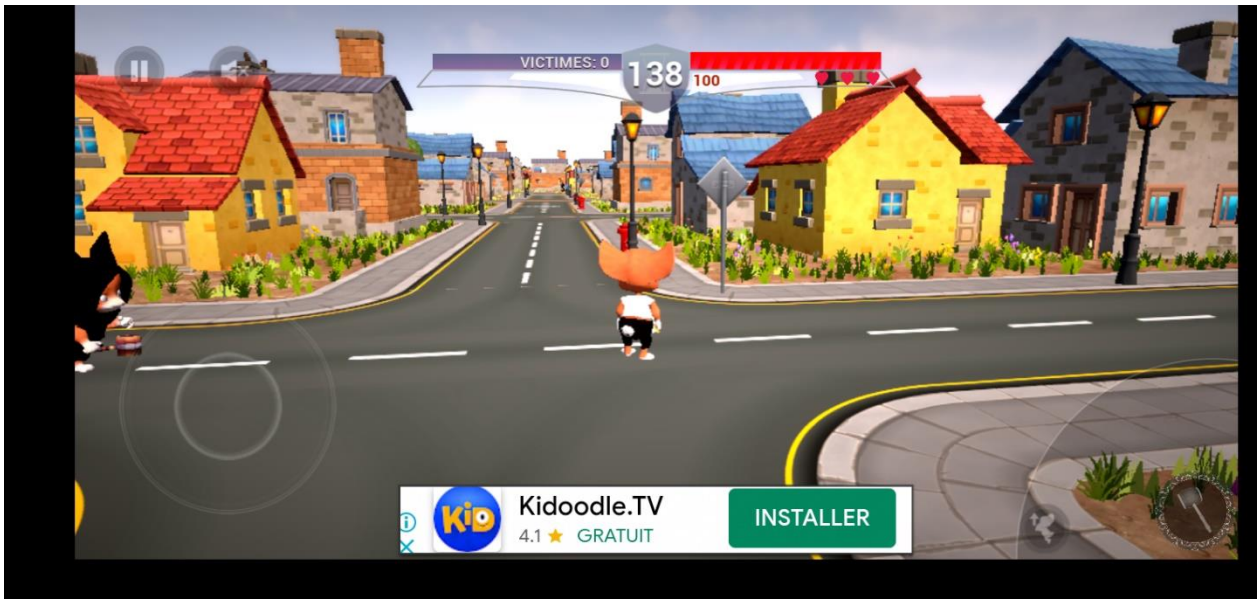


Figure V-15. Capture d'écran d'un bloc d'annonces de type bannière.

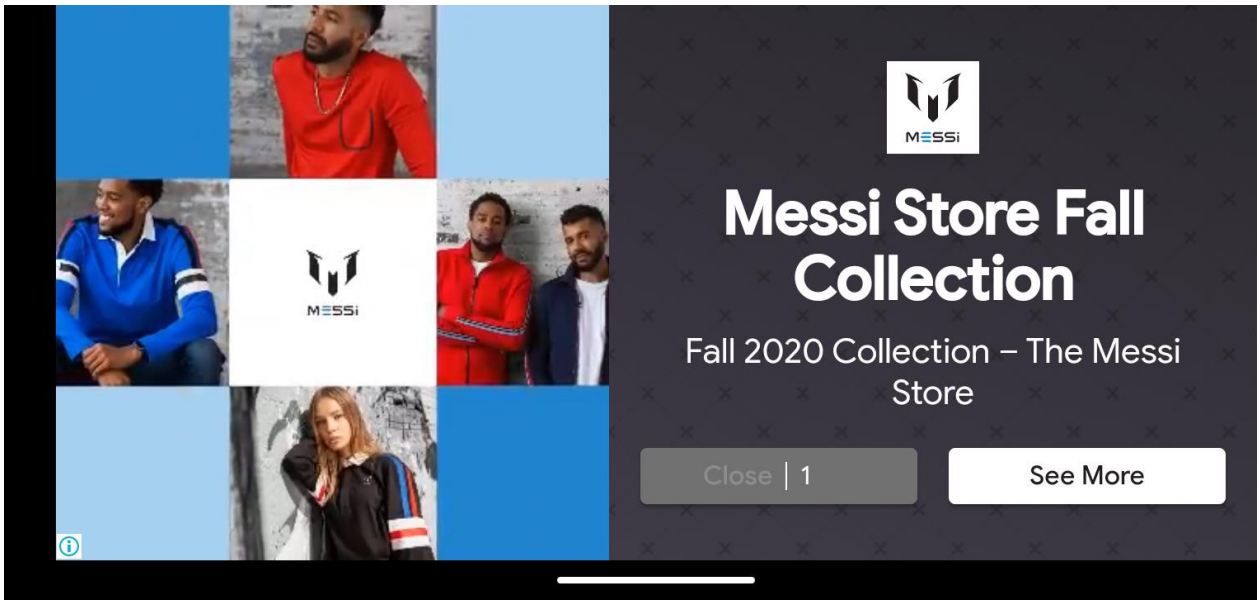


Figure V-16. Capture d'écran d'un bloc d'annonces de type interstitiel.

Remarque

- Ils existent des systèmes de publicités alternatives à Google AdMob tels que Chartboost et Unity.
- 30% de commissions sont prélevées par Google sur les revenues générées par le système AdMob.
- 30% de commissions sont prélevées par Google ou Apple sur les Achats intégrés '*In-App*' des applications disponibles sur leurs stores.
- L'intégration native des publicités AdMob dans Unreal n'est plus à jour, des plugins disponibles dans le store d'Epic sont nécessaires pour une bonne intégration des publicités dans les projets.

2. Publication sur le Play Store de Google

Pour pouvoir publier des jeux sur le Play Store de Google il faut suivre les étapes suivantes :

1. Créer une Key Store.
   - ➢ Une keystore est une clé unique qui garantira que projet sera lié au compte Google Play d'une seule personne et à personne d'autre.
   - ➢ Ouvrir l'invite de commande sur Windows.

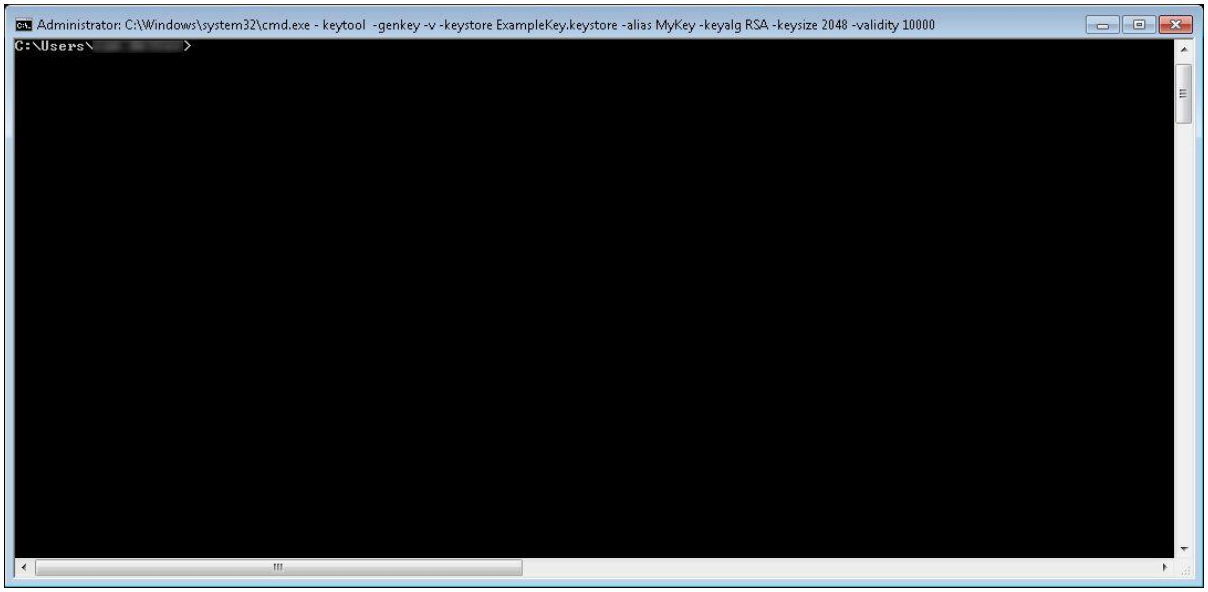


Figure V-17. Exemple de fenêtre d'invité de commande.

   - ➢ Taper la ligne de code suivante : *keytool-genkey -v -keystore ExampleKey.keystore -alias MyKey -keyalg RSA -keysize 2048 -validity 10000* ou ExampleKey et MyKey doivent être changé.
   - ➢ Suivre les instructions qui suivent comme : saisir le code de chiffrement, le nom, prénom, adresse et autres.

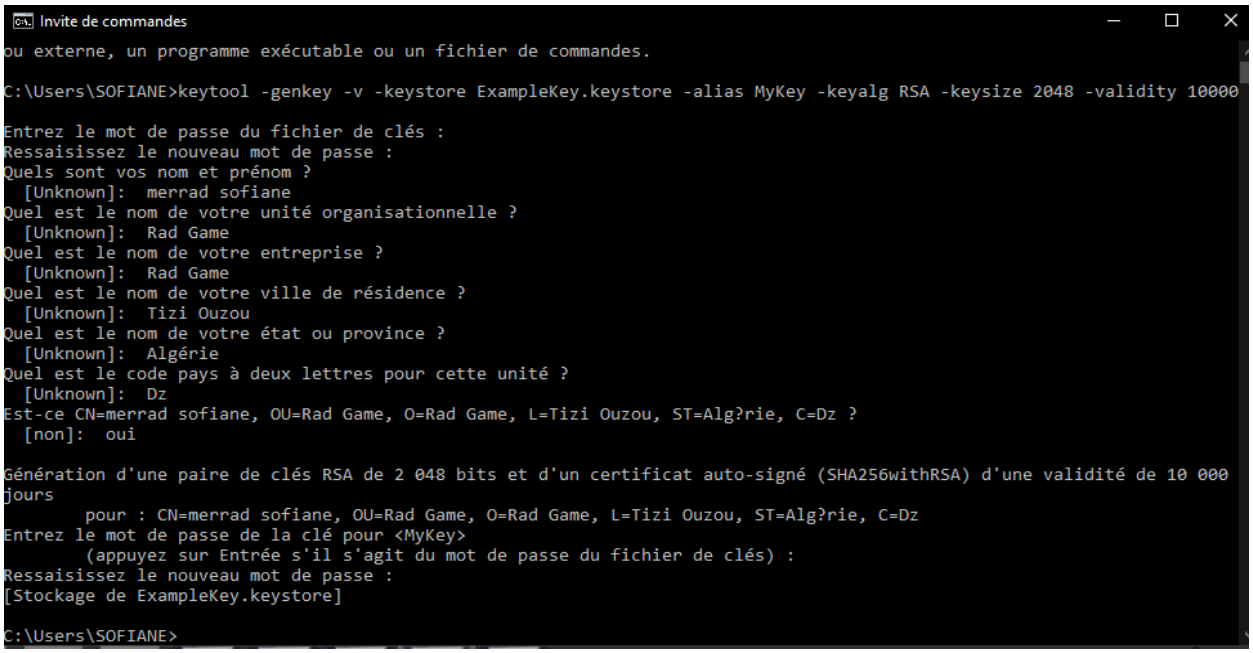


Figure V-18. Formulaire du keystore.

➢ La clé est stockée dans le dossier c:\user\current user et doit être copier dans le dossier du jeu dans le répertoire :

(Your Game Folder)\Build\Android.

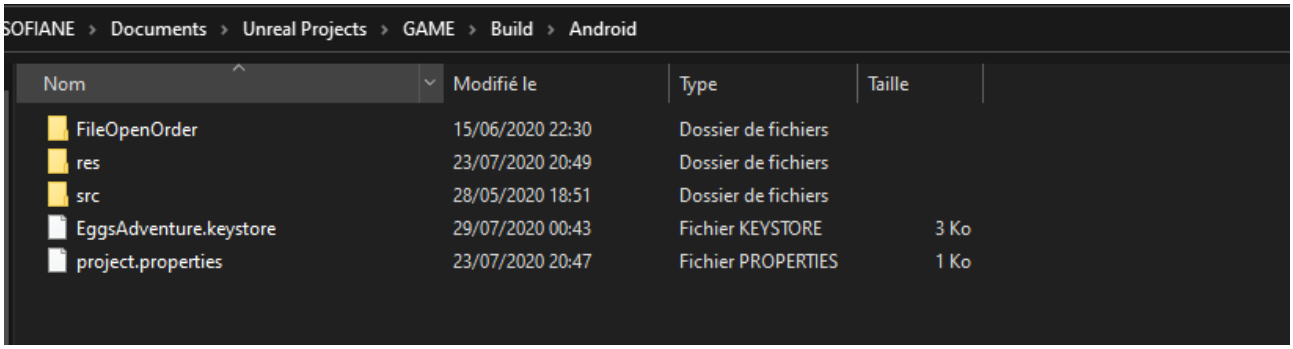


Figure V-19. Emplacement final de la Keystore.

➢ Aller au menu réglage du projet d'Unreal, dans la section Android puis Distribution Signing et saisir les informations demandées.

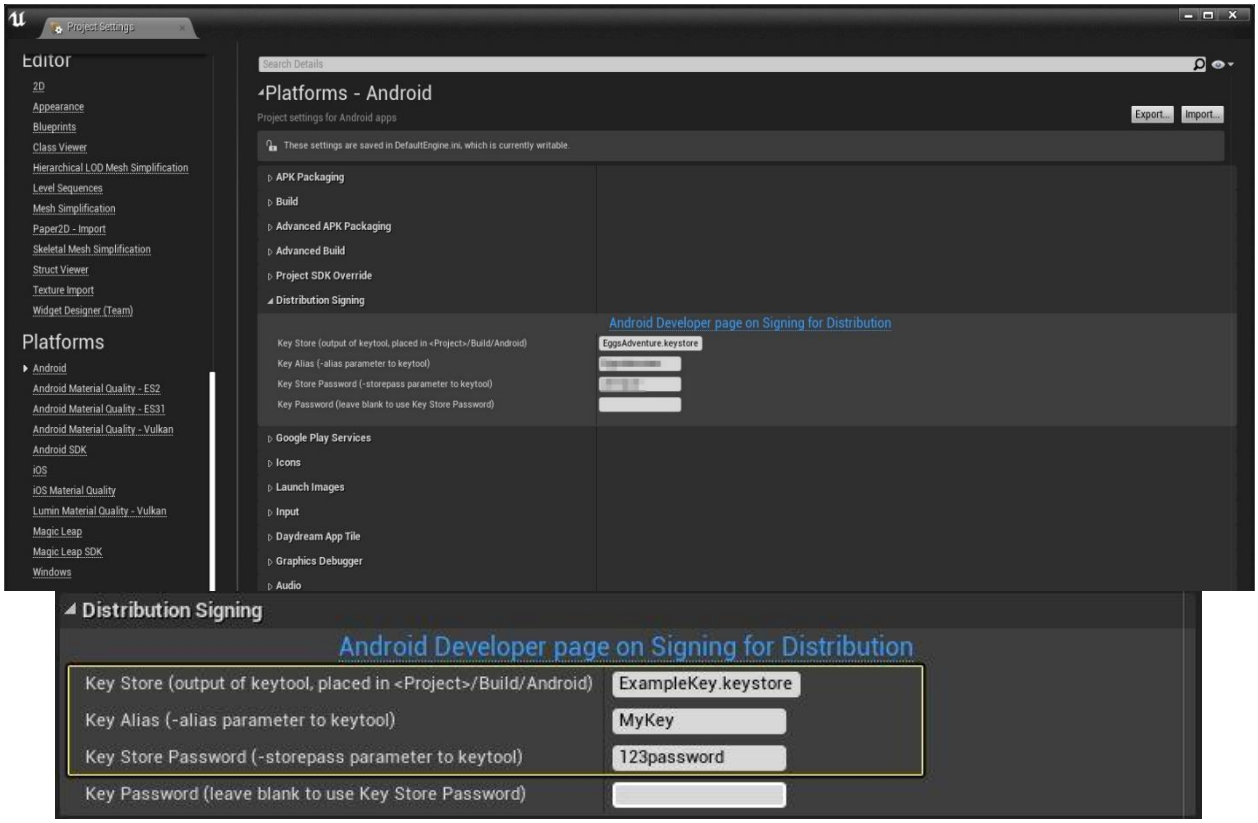


Figure V-20. Saisie des informations de la clé dans le moteur Unreal.

➢ Avant de fermer l'invite de commande, saisir la ligne de code : *keytool -exportcert -alias ExampleKey -keystore ExampleKey.keystore -list -v* ou on remplace ExampleKey et ExampleKey.keystore par les valeurs utilisées précédemment.

➢ Récupérer l'empreinte du certificat SHA1 et la stocker pour une utilisation ultérieure.

```
C:\Users\SOFIANE>keytool -exportcert -alias MyKey -keystore ExampleKey.keystore -list -v
Entrez le mot de passe du fichier de clés :
Nom d'alias : MyKey
Date de création : 3 oct. 2020
Type d'entrée : PrivateKeyEntry
Longueur de chaîne du certificat : 1
Certificat[1]:
Propriétaire : CN=merrad sofiane, OU=rad game, O=rad gam, L=tizi ouzou, ST=alg?rie, C=dz
Emetteur : CN=merrad sofiane, OU=rad game, O=rad gam, L=tizi ouzou, ST=alg?rie, C=dz
Numéro de série : ab2909e
Valide du : Sat Oct 03 17:14:03 WAT 2020 au : Wed Feb 19 17:14:03 WAT 2048
Empreintes du certificat :
         MD5:  42:45:8E:93:A0:8E:02:0B:38:C3:B7:8F:34:40:E8:98
         SHA1 : F5:A7:A0:0C:F5:EE:FC:3B:10:66:FF:F0:45:06:36:F5:01:38:EF:2C
         SHA256: F2:87:87:77:79:9D:7E:3E:94:E6:CF:8F:8E:2E:91:4B:66:8C:14:64:E6:81:B0:B6:DF:81:D4:5E:FA:76:4A:74
         Nom de l'algorithme de signature : SHA256withRSA
         Version : 3
```

Figure V-21. Certificat SHA1 à conserver.

Remarques

- La keystore permet de certifier d'une manière unique une application.
- Toutes les applications publiées sur le store de Google doivent être signées.
- La keystore permet de s'assurer que les mises à jour d'une application sont effectuées par la même personne et non pas par un intrus.
- En cas de perte du keystore, les mise à jour d'une application ne seront plus possibles et le développeur doit créer une nouvelle application dans la console de google avec un nom de package différent.

2. Créer un compte développeur chez Google sur le site '*https://developer.android.com/distribute/console*'.

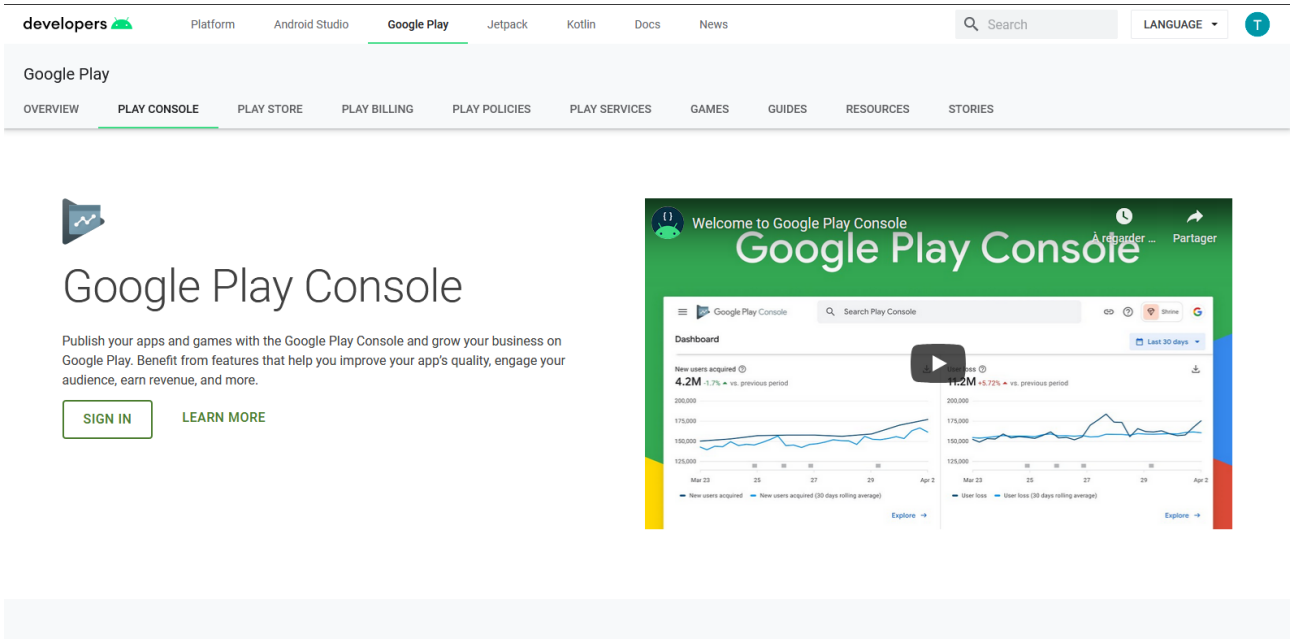


Figure V-22. Page d'accueil de la console Google Play.

3. S’inscrire sur le site.

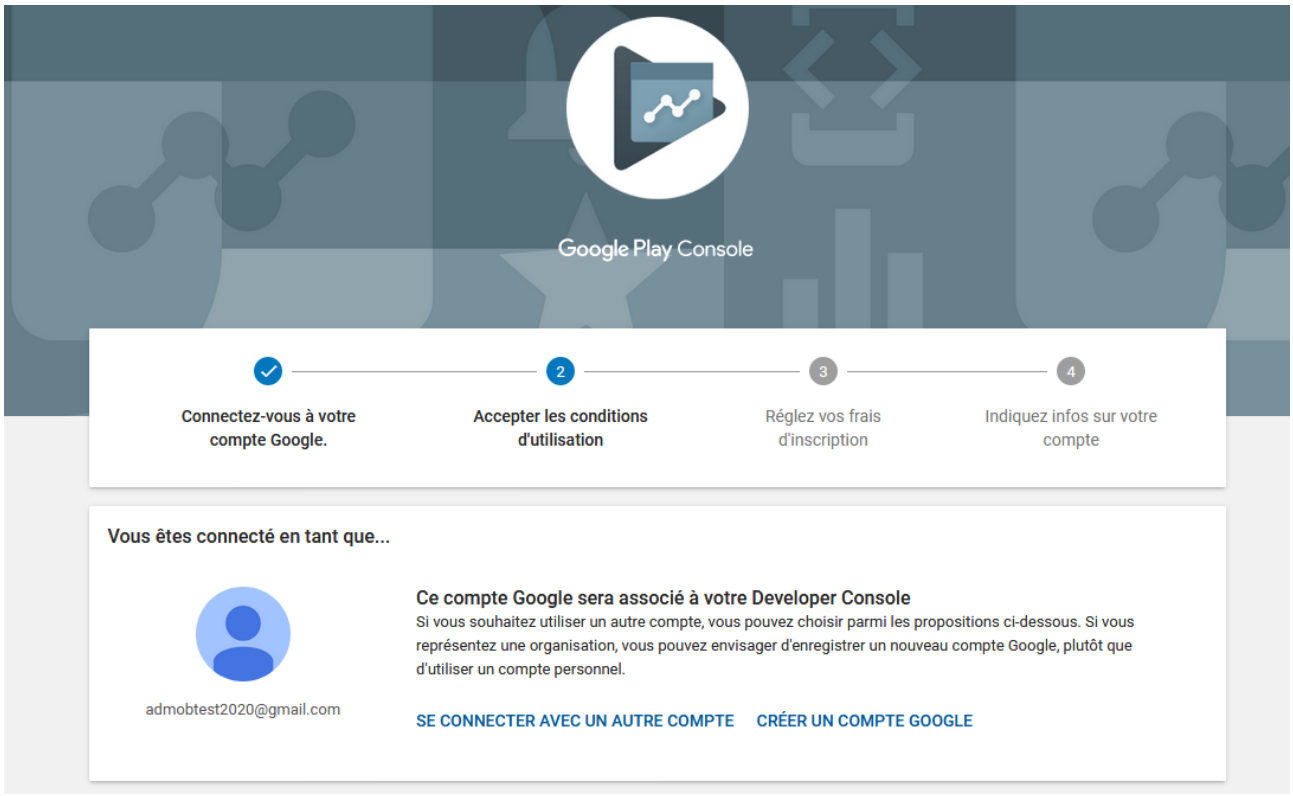


Figure V-23. Etapes d’inscription sur Google Play Console.

4. Payer 25 $US ou équivalent pour pouvoir utiliser les services de Google Play Console.

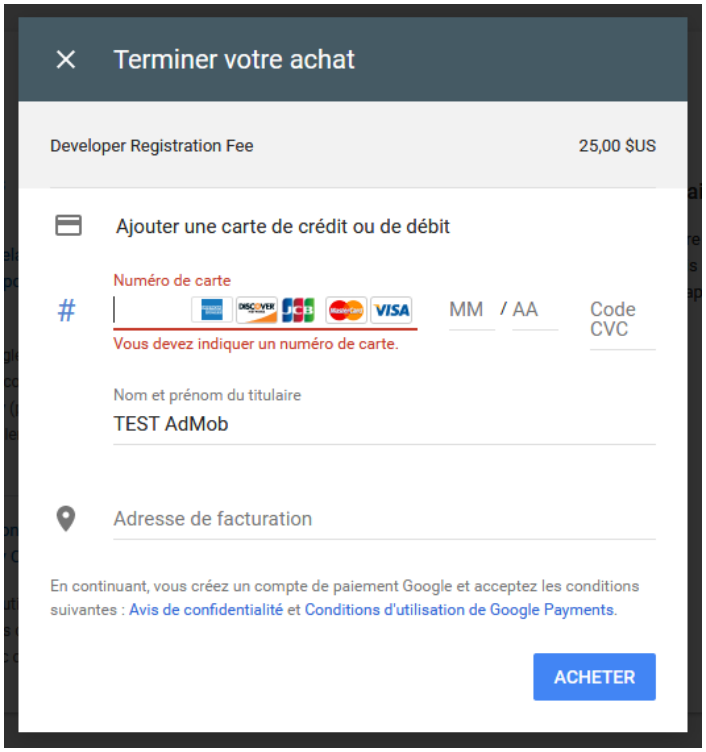


Figure V-24. Formulaire de payement du compte.

5. Après la finalisation de l'inscription, la page d'accueil sera affichée.

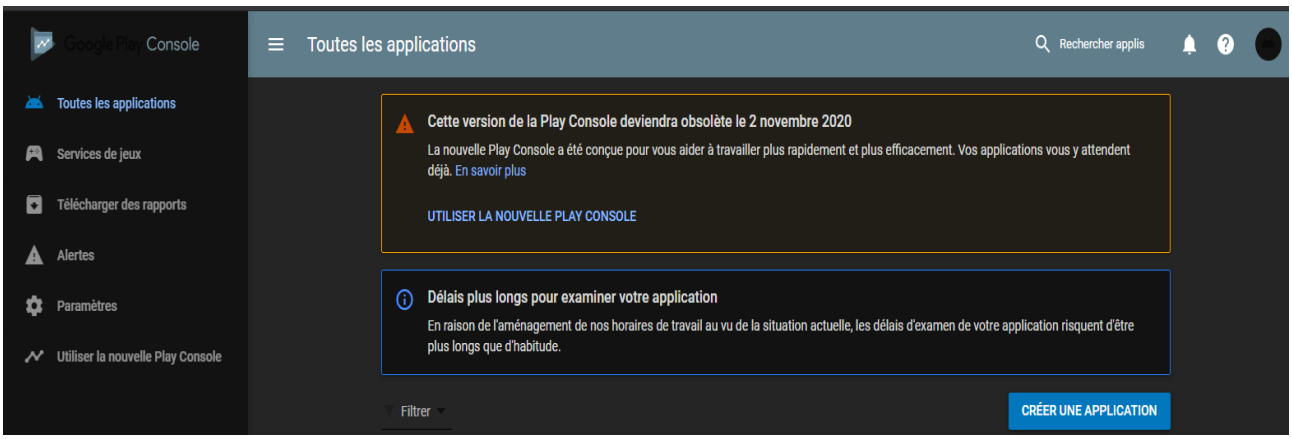


Figure V-25. Page d'accueil de la console Google Play.

6. L'étape suivante est la création d'une application à l'aide du bouton 'CREER UNE APPLICATION'.

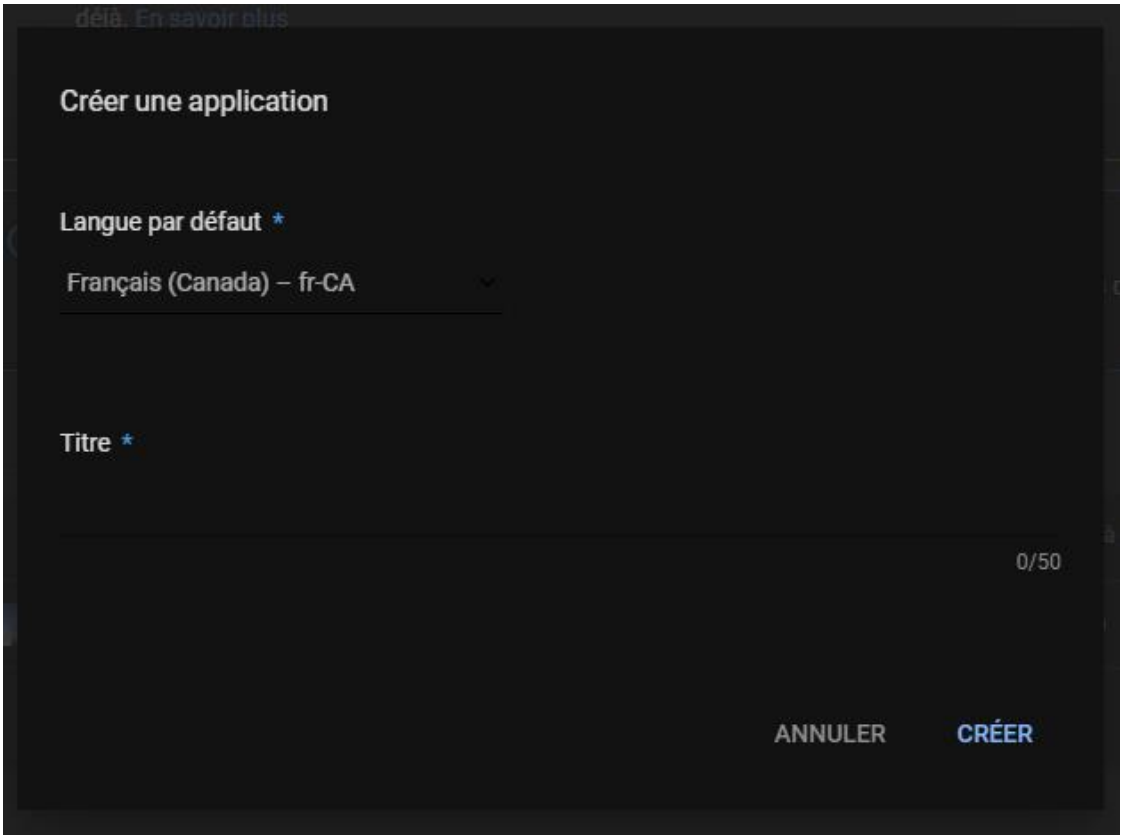


Figure V-26. Formulaire à remplir pour créer une application sur la console.

7. Après la création de l'application beaucoup d'informations doivent être saisie pour valider l'application
   - ➢ La saisie des champs de la fiche de présentation du jeu sur le Play Store comme le titre, une description, des captures d'écran du jeu, des icônes.

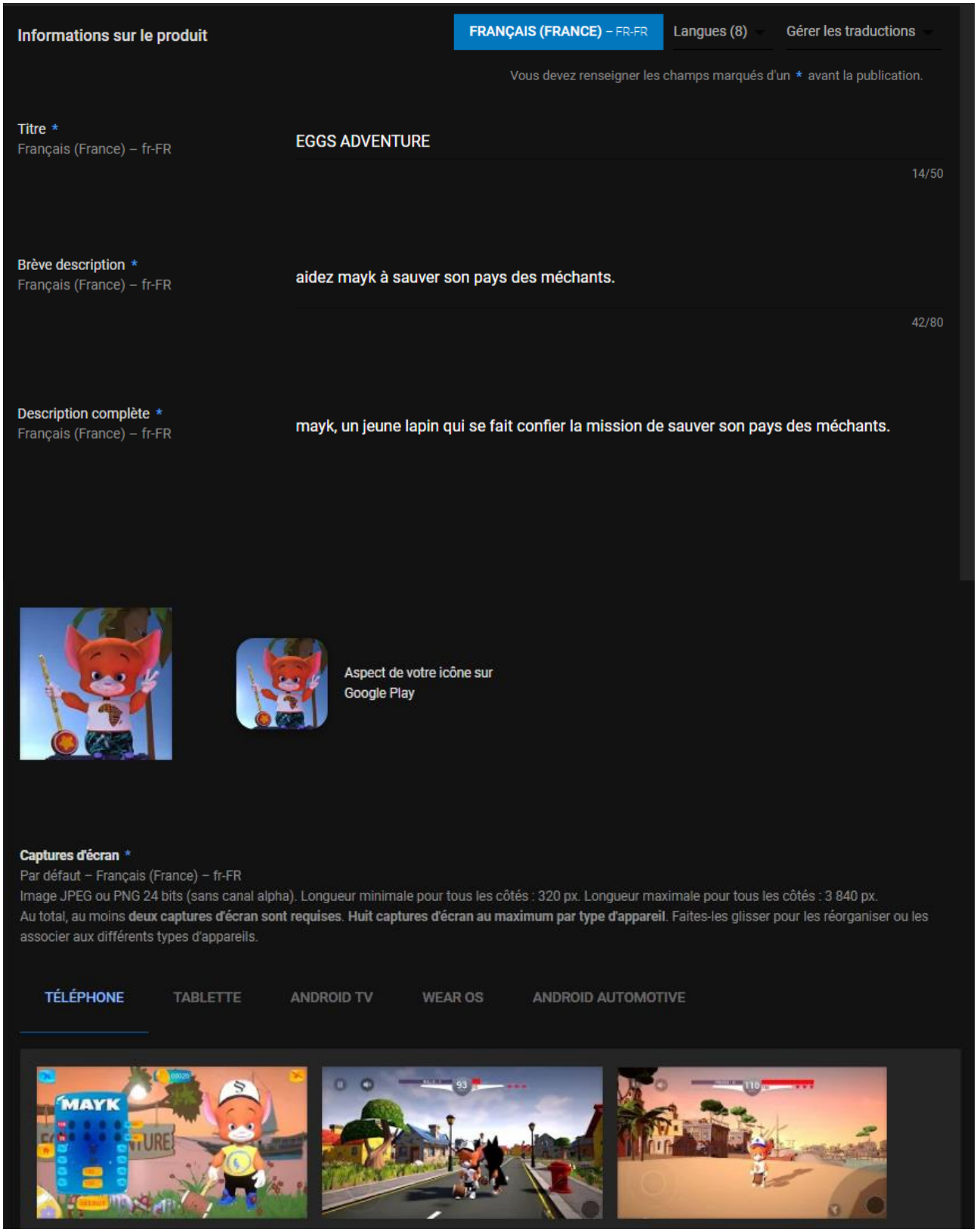


Figure V-27. Fiche de présentation du jeu EGGS ADVENTEURE.

- Gérer le tarif et la disponibilité de l'application.

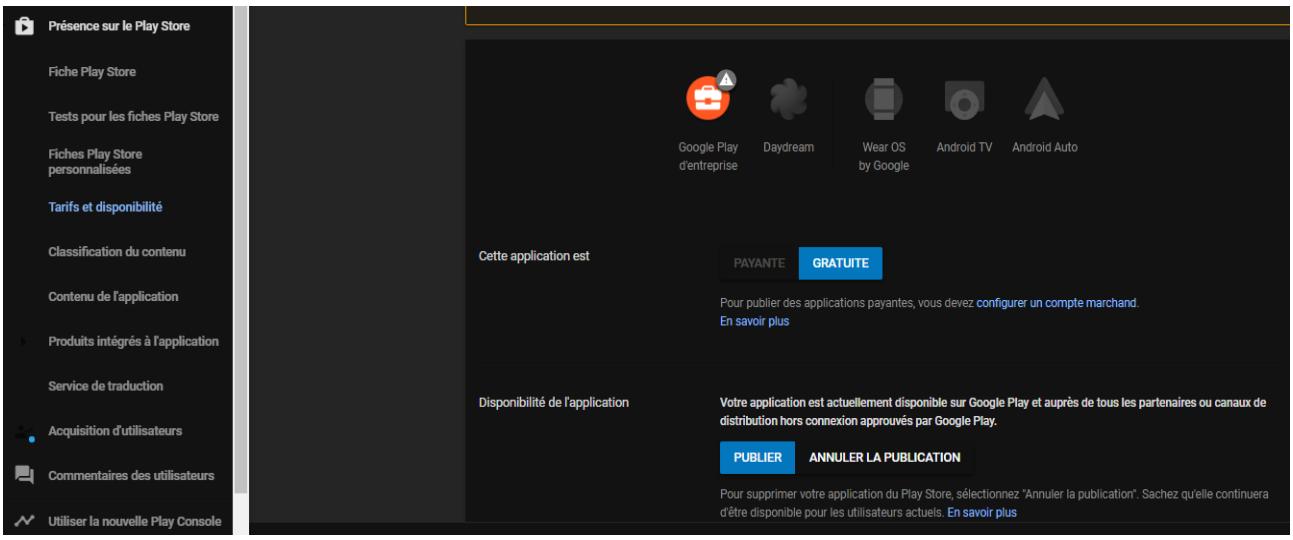


Figure V-28. Tarif et disponibilité du jeu EGGS ADVENTEURE.

- Décrire le contenu de l'application tel que les règles de confidentialité, les annonces, l'accès à l'application, classification du contenu du jeu et sa cible.

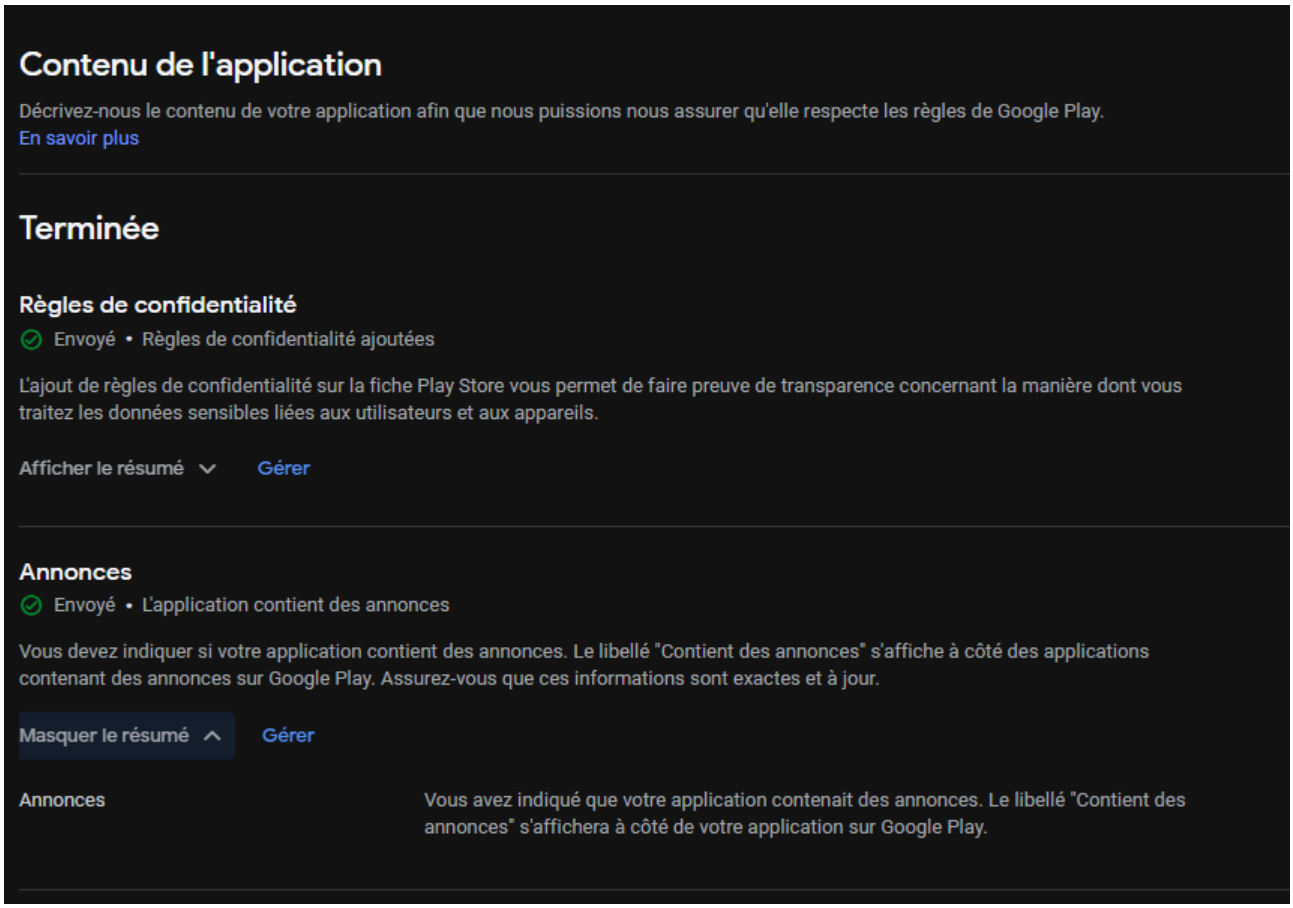


Figure V-29. Résumé des règles de confidentialité et annonces dans EGGS ADVENTURE.

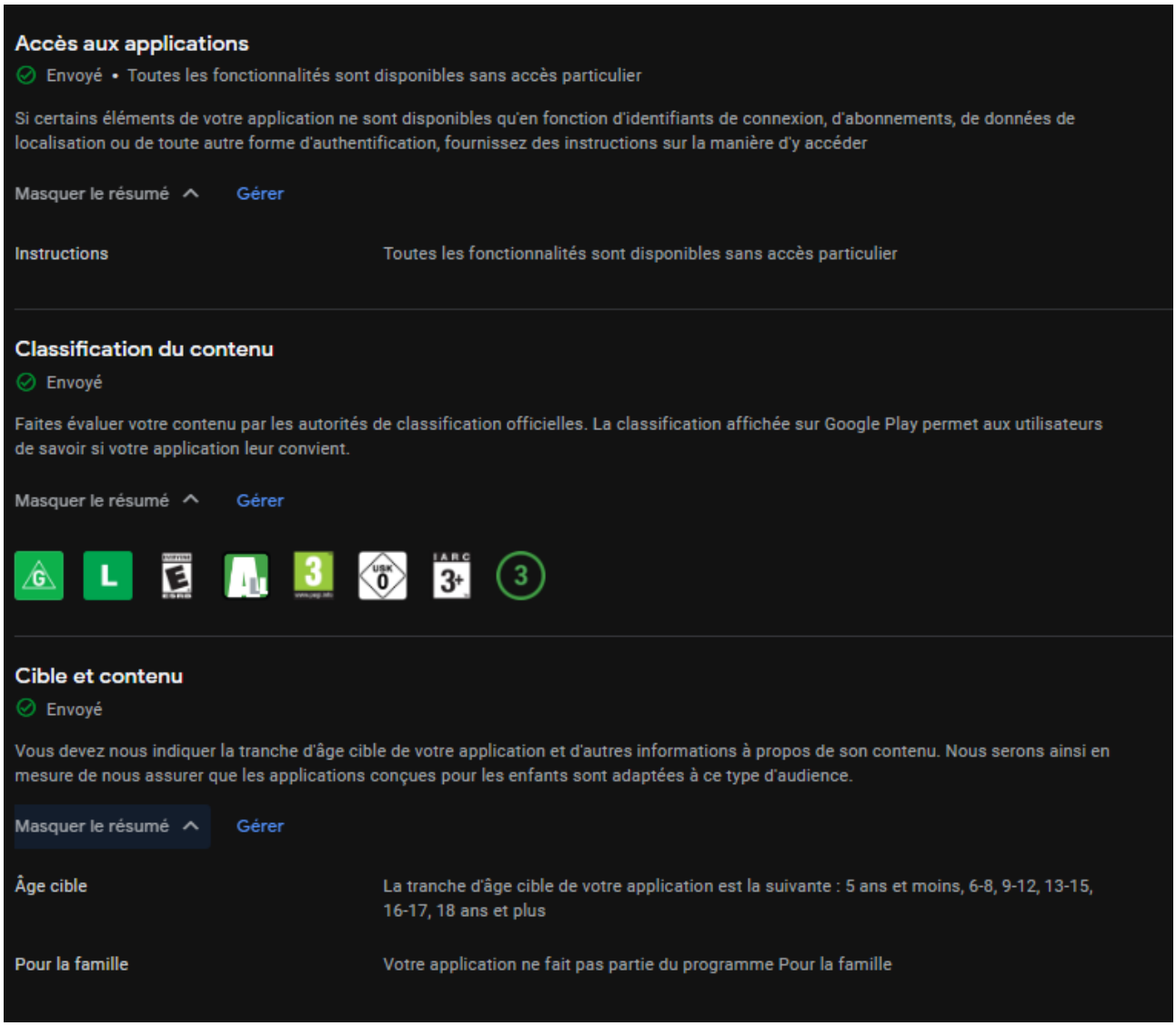


Figure V-30. Résumé des règles d'accès aux applications, classification et cible dans EGGS ADVENTURE.

8. Récupérer la clé de licence qui permet d'empêcher toute distribution non autorisée de votre application. Elle peut également être utilisée pour valider les achats facturés via l'application.

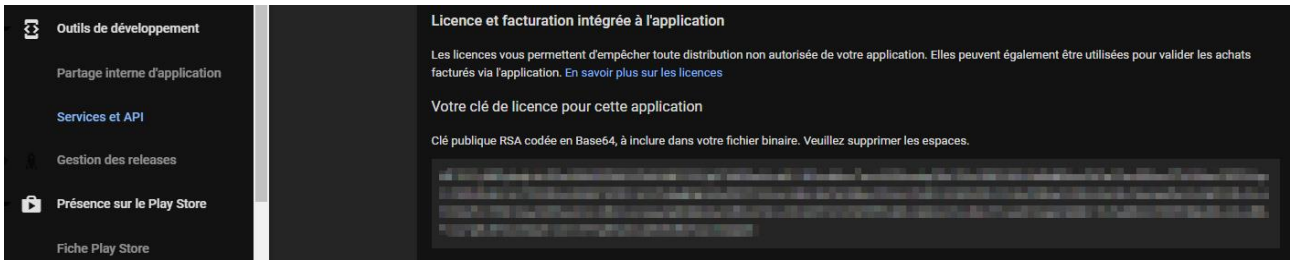


Figure V-31. Récupération de la clé de licence.

9. Saisir la clé de licence dans la partie Google Play Licence Key du sous menu Google Play Service du menu Android des réglages du projet.
    - La clé de licence est vérifiée à chaque démarrage d'une application payante pour vérifier l'achat de cette dernière par l'utilisateur et non son acquisition d'une manière illégale, c'est pour cela qu'une connexion internet est toujours requise pour ce genre d'application.

Figure V-32. Saisie de la clé de licence dans le projet.

10. Pour configurer les services de jeux google Play comme le leaderboard, les succès et autre
    - Aller dans le menu service de jeux dans la console Google Play.
    - Aller au sous menu applications associées.

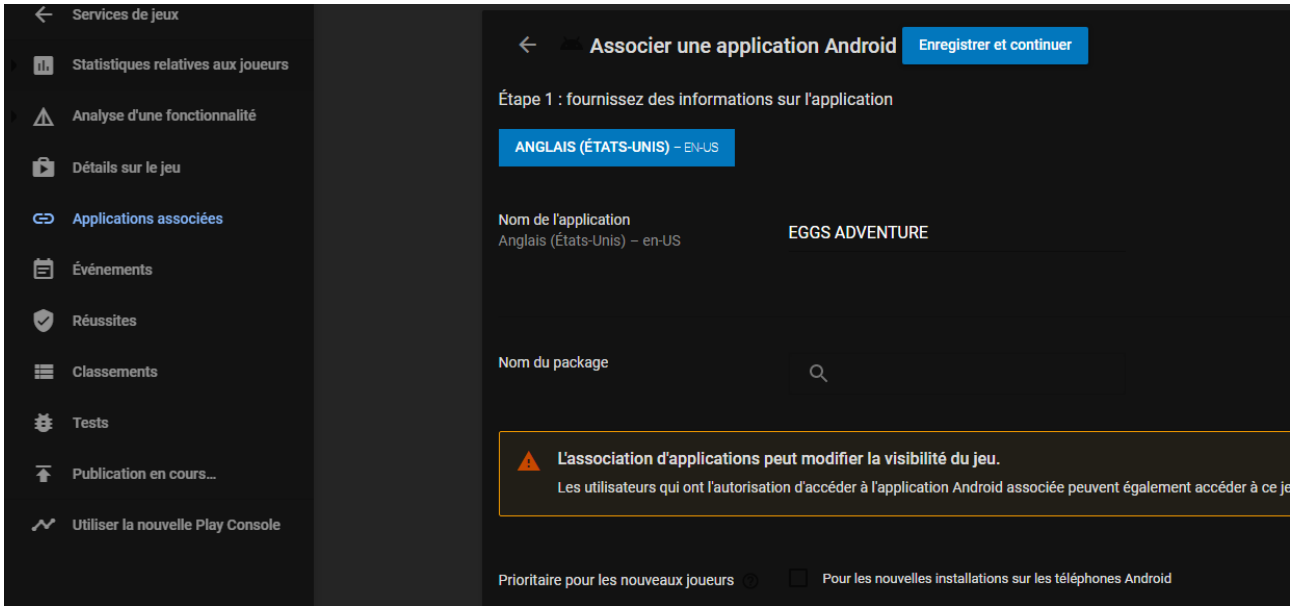


Figure V-33. Menu application associées dans les services de jeux.

- Saisir le nom du package contenu dans les réglages du projet dans la section Android.
- Appuyer sur le bouton Enregistrer et continuer.
- Saisir le SHA1 précédemment copié dans le champ SHA1 qui s'affiche.

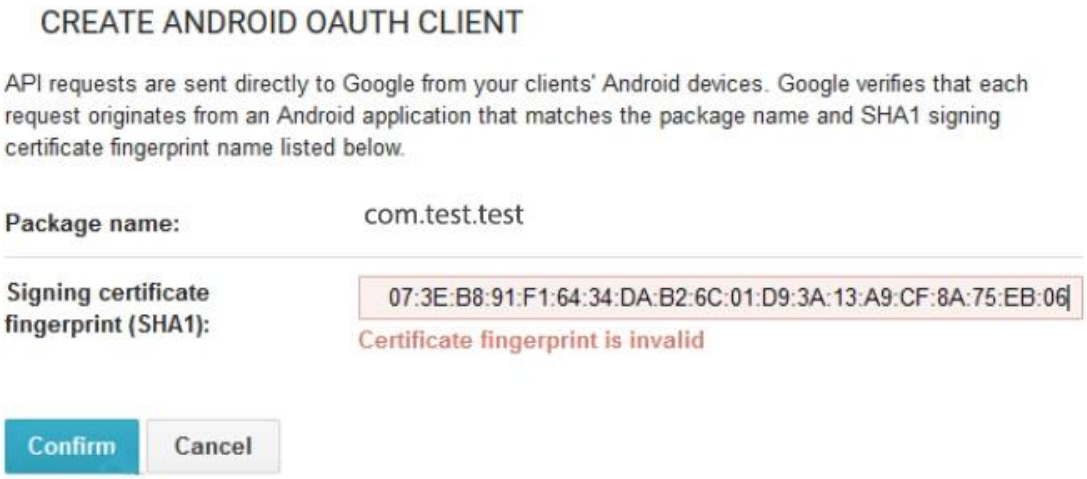


Figure V-34 Saisie du certificat SHA1.

- Récupération d'un Identifiant de l'application dans la fenêtre qui s'affiche et le copier dans Game App Id du sous menu Google Play Service du menu Android des réglages du projet.

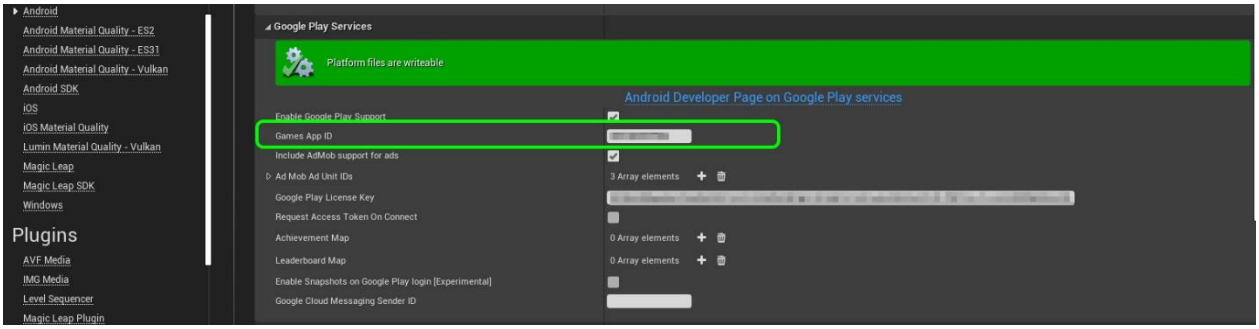


Figure V-35. Saisie de l'identifiant de l'application dans le projet.

11. Compiler le jeu pour la plateforme Android dans Unreal en choisissant les configurations suivantes :

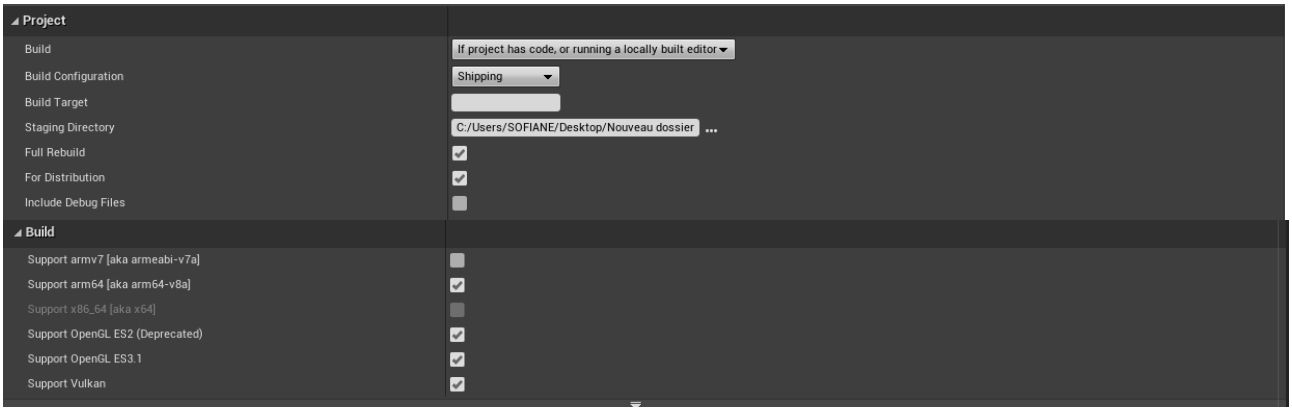


Figure V-36. Exemple de configuration destiné aux mobiles ARM64 supportant L'OpenGL ES2, ES3.1 et Vulkan [16].

12. Dans le menu outils de développement, le sous menu Releases de l'application permet de charger le fichier APK ainsi que le fichier d'extension OBB généré par le moteur.

Figure V-37. Fenêtre de gestions des release de l'application.

- Les versions Alpha et Beta permettent de distribuer l'application pour un nombre limité de testeurs choisis au préalable.
- La version Production lance l'application pour le grand public dans le Play Store.

13. Dans le menu production, le développeur peut ajouter les fichier APK et OBB pour une nouvelle distribution de son jeu.

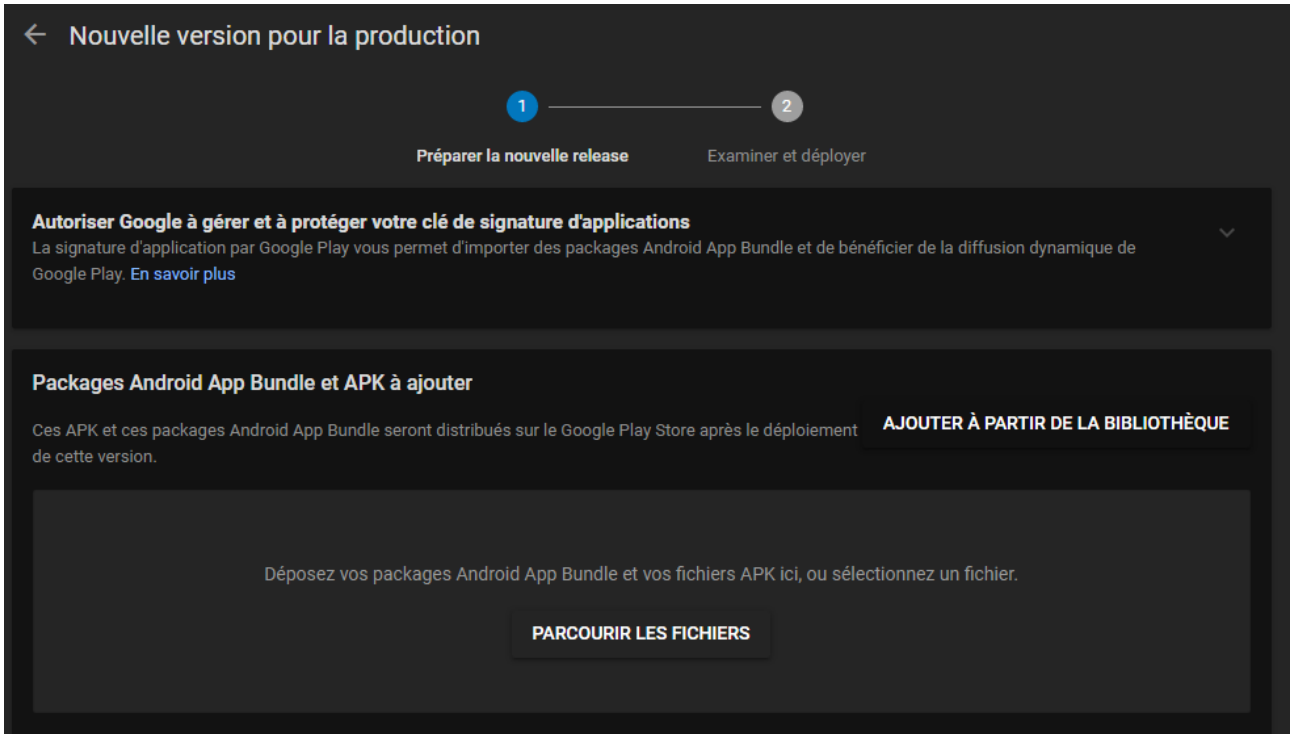


Figure V-38. Déploiement d'une nouvelle version du jeu.

14. Après le téléchargement vers le serveur du jeu, un nom de version et la saisie des nouveautés sont nécessaires.

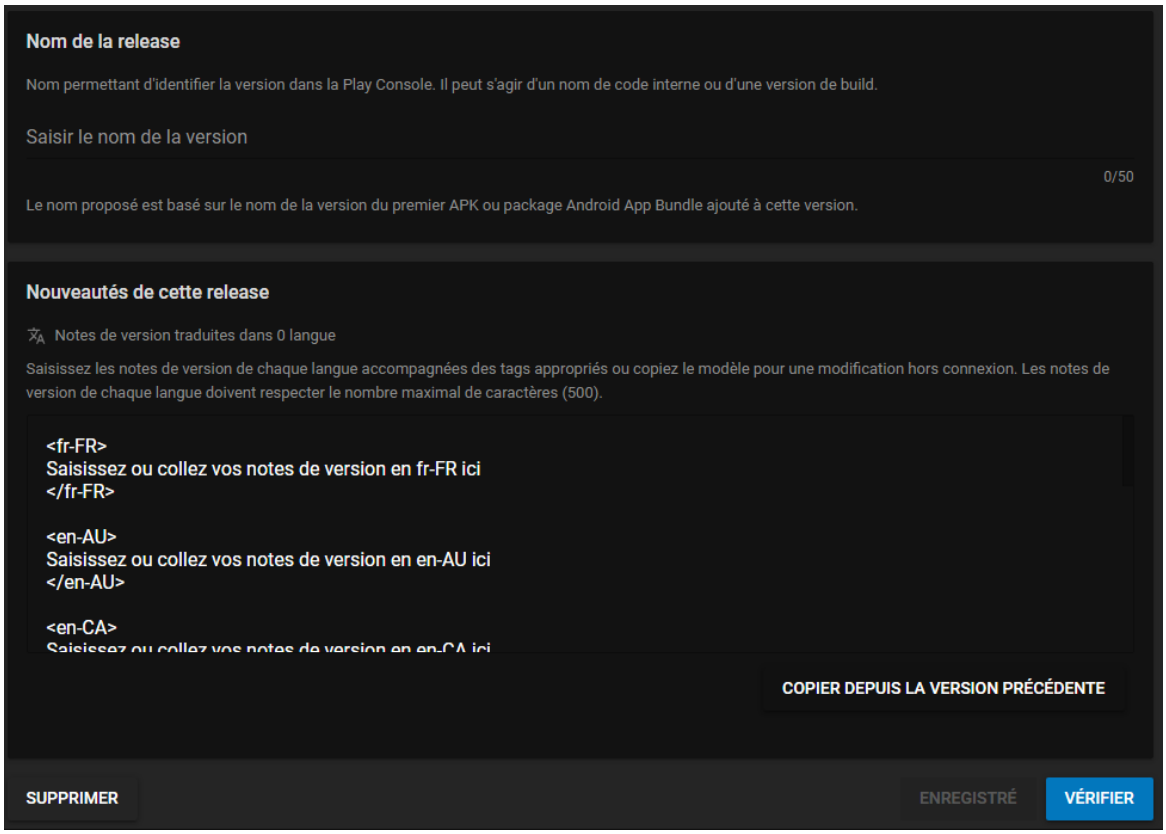


Figure V-39. Saisie du nom de release et des nouveautés.

15. Le bouton ‘vérifier’ vérifie d’éventuelle erreurs. A la fin de la vérification un bouton ‘déployer’ sera afficher pour déployer l’application sur le store qui prendra effet au bout de quelques jours.

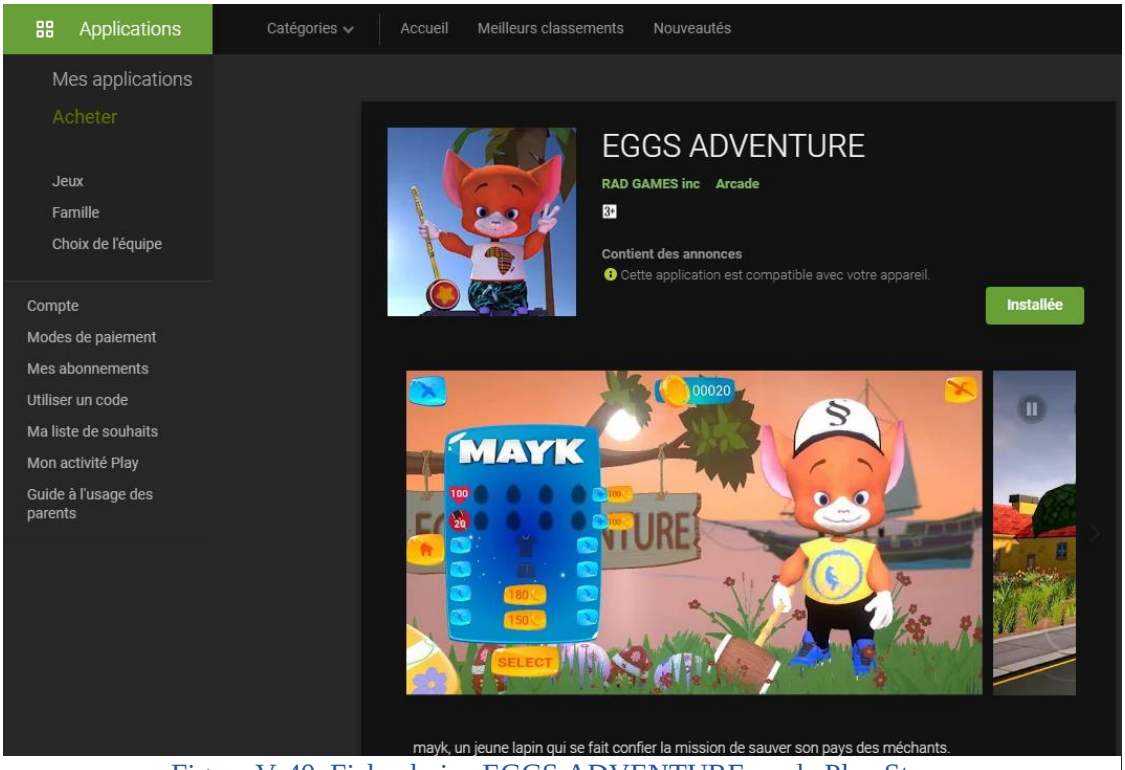


Figure V-40. Fiche du jeu EGGS ADVENTURE sur le Play Store.

Adresse de téléchargement de EGGS ADVENTURE

https://play.google.com/store/apps/details?id=com.MERRADORG.EGGSADVENTURE

Figure V-41. QR code de l'adresse de téléchargement.

3. Conclusion

Dans ce chapitre, nous avons détaillé les différentes étapes de la création d'un compte AdMob et d'intégration de publicités dans le jeu, ensuite nous avons expliqué le processus de la publication de jeu sur le Play store de GOOGLE.

Conclusion générale

L'objectif de notre travail était de réaliser un jeu pour la plateforme mobile Android avec le moteur de jeu Unreal Engine et par cette démarche nous sommes arrivés à avoir une idée générale sur le fonctionnement de ce dernier, et on a eu la possibilité de connaitre ses caractéristiques et ses fonctionnalités et le rôle de chacune.

La réalisation du jeu EGGS ADVENTURE a consisté à appliquer toutes nos connaissances dans le domaine de l'infographie et de la création de jeu pour pouvoir réaliser un jeu aux normes actuelles qui sera destiné au grand public dans des délais réduits.

Le développement de ce jeu nous a permis d'acquérir des connaissances supplémentaires dans le domaine de l'informatique (théorique et pratique) et spécialement dans le domaine de l'infographie. Le travail présenté nous a permis de :

- D'approfondir nos connaissances sur les techniques de l'infographie et des jeux vidéo.
- De se familiariser avec les méthodes utilisées dans l'industrie du jeu pour la conception d'un jeu et la création de d'un cahier de conception.
- D'accroitre notre maitrise des outils de développement graphique comme Adobe Photoshop, Adobe Illustrator, Autodesk 3DS Max, Autodesk Motion Builder.
- D'étudier en détail le fonctionnement du moteur de jeu Unreal Engine qui est l'un des standards actuels de la profession.
- D'utiliser les outils et fonctionnalités qu'offre le moteur pour développer, réaliser, optimiser et publier notre jeu.
- Enfin, de maitriser les outils et techniques nécessaires à l'intégration de publicité dans notre jeu et de sa publication sur un magasin en ligne.

Chaque jeu est sujet d'amélioration et de perfectionnement, et dans notre cas les perspectives de notre jeu consistent à :

- Ajouter d'autres personnages et d'améliorations.
- Améliorer le game play et l'expérience de jeu.
- Déployer le jeu sur d'autres plateformes comme Windows, IOS et Mac.
- Intégrer un système de multijoueur.

Enfin, espérons que notre travail sera d'une utilité à toute personne intéressée par ce sujet et que ce type d'application prenne d'avantage d'importance dans notre pays.

Webographie

[1] https://whatis.techtarget.com/fr/definition/modelisation-3D

[2] https://info.e-onsoftware.com/home

[3] https://store.speedtree.com/

[4] https://www.unchartedthegame.com/fr-fr/

[5] https://www.develop4fun.fr/lhistoire-des-jeux-video-introduction/

https://fr.wikipedia.org/wiki/Cathode-ray_tube_amusement_device

[6] https://www.afjv.com/news/10047_jeux-video-mobiles-86-milliards-de-depenses-en-2019.htm

[7] http://www.csc.kth.se/utbildning/kth/kurser/DH2640/grip08/HighConceptTemplate-Inl4.pdf

[8] https://www.google.com/intl/fr_fr/adsense/start/

[9] https://www.e-tribart.fr/blog/secteur-3d/logiciels-techniques/4-moteurs-de-jeux

[10] https://www.mixamo.com/

[11] https://www.autodesk.com/products/3ds-max/overview

[12] https://www.adobe.com/

[13] https://www.epicgames.com/store/fr/

[14] https://bethesda.net/en/game/doom

[15] https://fr.wikipedia.org/wiki/AdMob

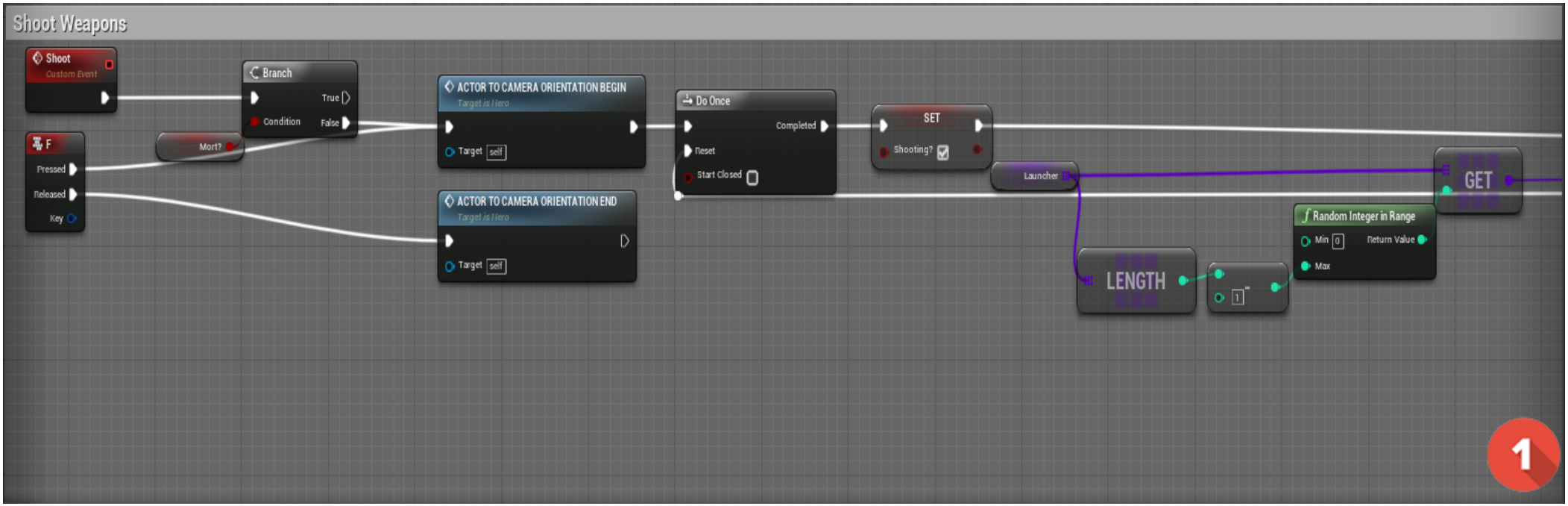
[16] https://www.khronos.org/vulkan/

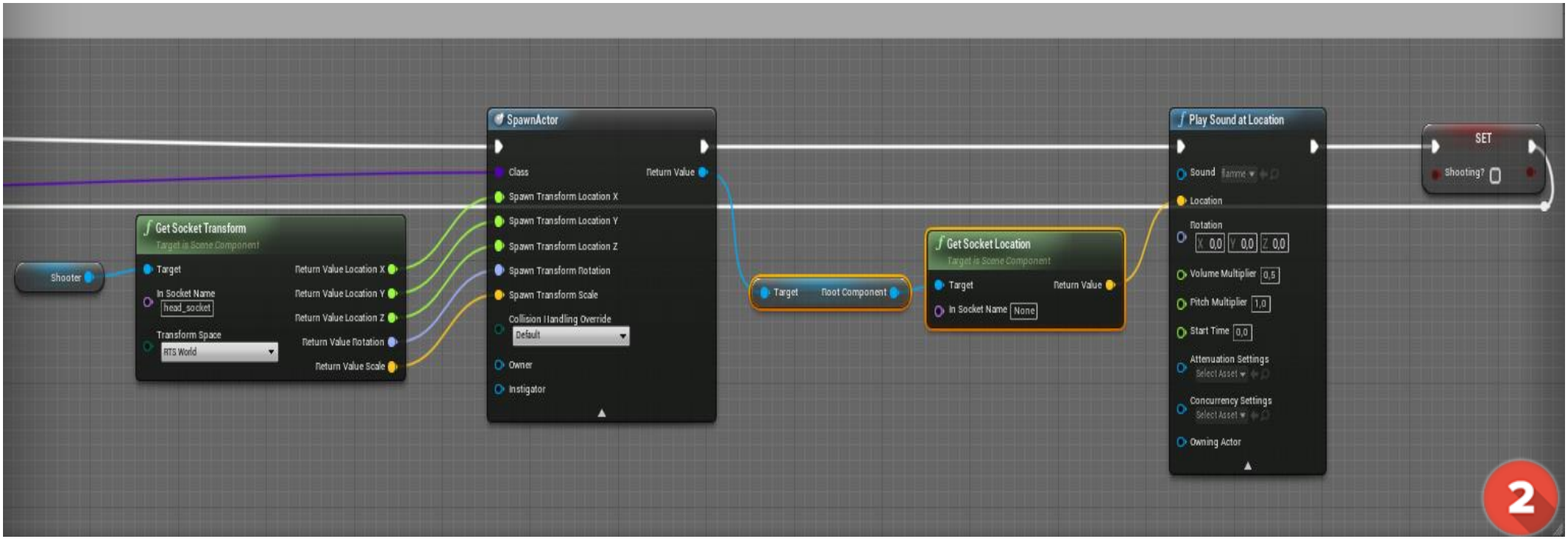
Bibliographie

1. Amirouche Fatiha (2018). Cours d'infographie, (3ème année licence, université Mouloud Mammeri.)
2. Ramakrishnan Mukundan (2012). Advanced Methode in Computer Graphics.
3. Dan Ginsburg et Budirijanto Purnomo (2014). OpenGL ES 3.0 Programming Guide.
4. Peter L. Newton et Jie Feng (2016). Unreal Engine 4 AI Programming Essentials.
5. Brenden Sewell (2015). Blueprints Visual Scripting for Unreal Engine.
6. Jesse Schell (2008). The Art of Game Design.
7. Randi L. Derakhshani, Dariush Derakhshani (2014). Autodesk 3ds Max 2015 Essentials : Autodesk Official Press.
8. Grégory Gosselin De Bénicourt (2015). Les cahiers d'Unreal Engine T1 : Modélisation, Blueprints, Matériaux et Paysages.
9. Grégory Gosselin De Bénicourt (2015). Les cahiers d'Unreal Engine T2 : Personnages, Intelligence Artificielle et Particules
10. Grégory Gosselin De Bénicourt (2015). Les cahiers d'Unreal Engine T3: Du Level Design au Packaging.

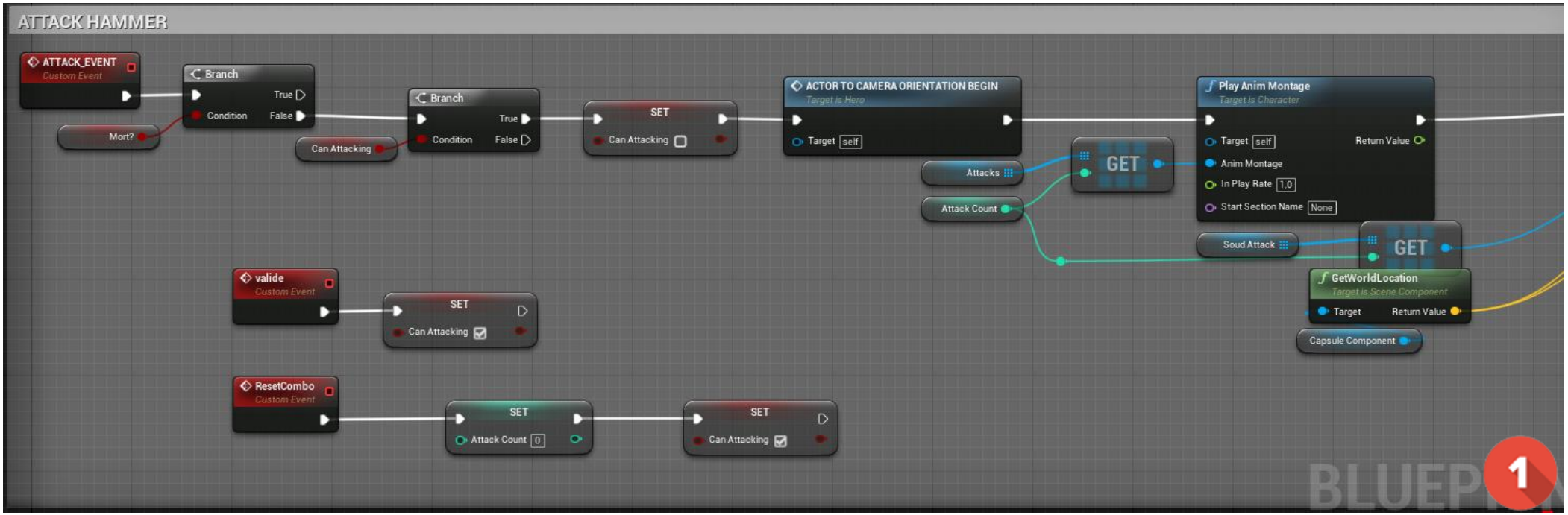
Annexes

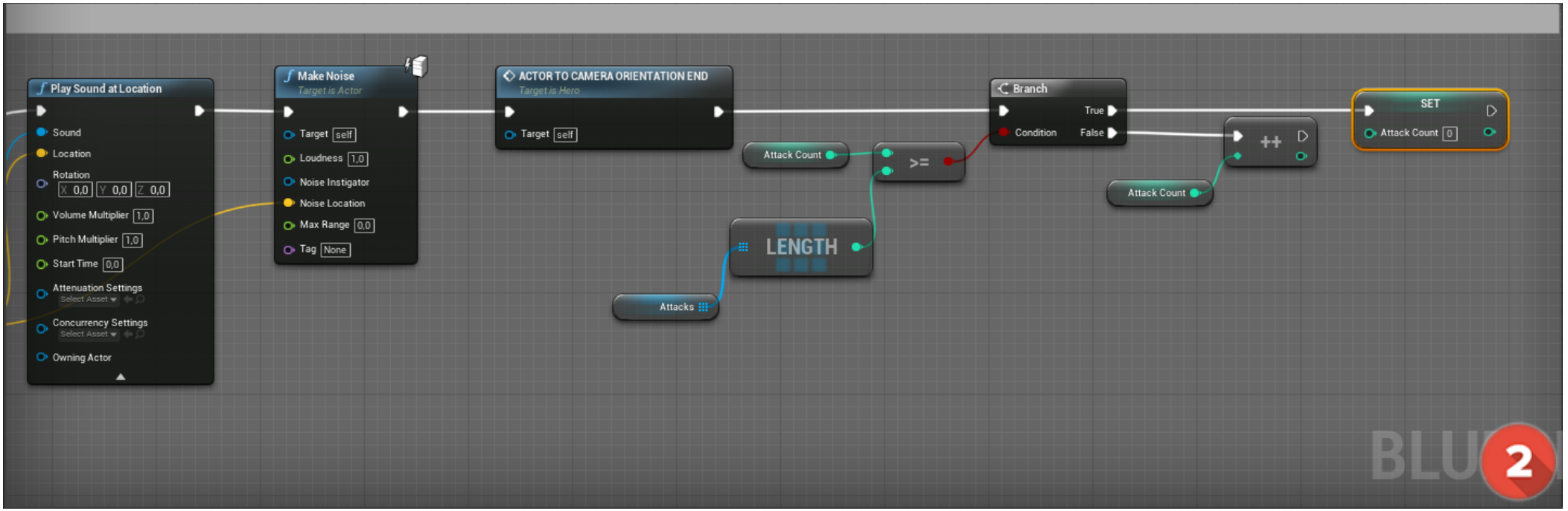
ANNEXE 1 : Blueprint attaquer avec bombe du héros.


Shoot Weapons
Shoot
Custom Event
Branch
True
Condition
False
Mort?
F
Pressed
Released
Key
ACTOR TO CAMERA ORIENTATION BEGIN
Target is Hero
Target self
ACTOR TO CAMERA ORIENTATION END
Target is Hero
Target self
Do Once
Completed
Reset
Start Closed
SET
Shooting?
Launcher
LENGTH
1
Random Integer in Range
Min 0
Max
Return Value
GET
1

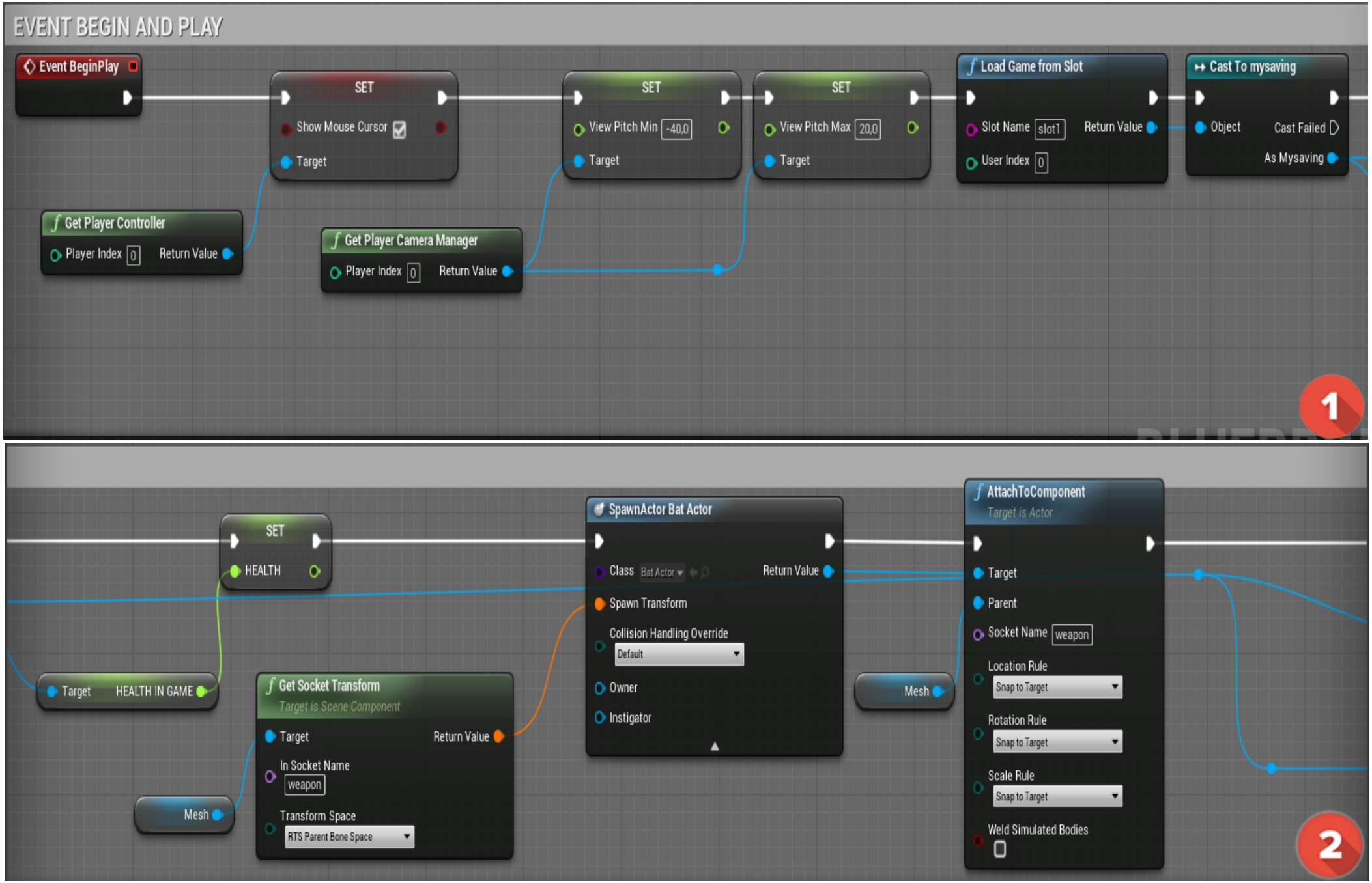

Shooter
Get Socket Transform
Target is Scene Component
Target
In Socket Name
head_socket
Transform Space
RTS World
Return Value Location X
Return Value Location Y
Return Value Location Z
Return Value Rotation
Return Value Scale
SpawnActor
Class
Spawn Transform Location X
Spawn Transform Location Y
Spawn Transform Location Z
Spawn Transform Rotation
Spawn Transform Scale
Collision Handling Override
Default
Owner
Instigator
Return Value
Target
Root Component
Get Socket Location
Target is Scene Component
Target
In Socket Name
None
Return Value
Play Sound at Location
Sound
Location
Rotation
X 0,0
Y 0,0
Z 0,0
Volume Multiplier
0,5
Pitch Multiplier
1,0
Start Time
0,0
Attenuation Settings
Concurrency Settings
Owning Actor
SET
Shooting?
2

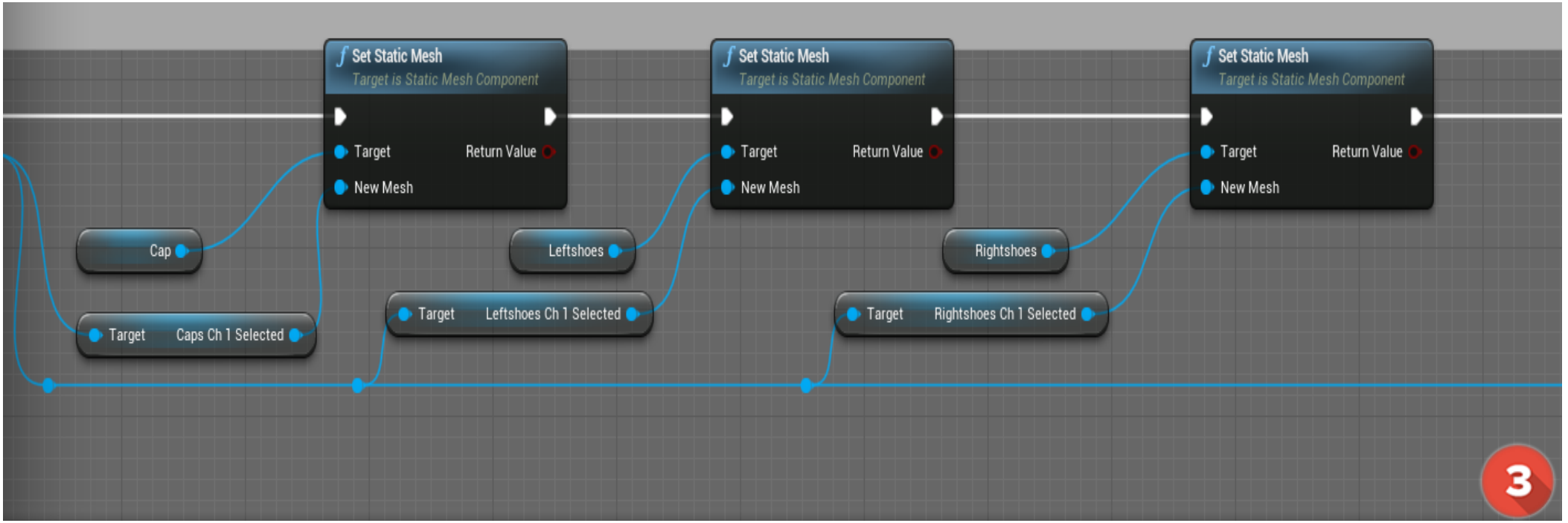
ANNEXE 2 : Blueprint attaque à la main du héros.

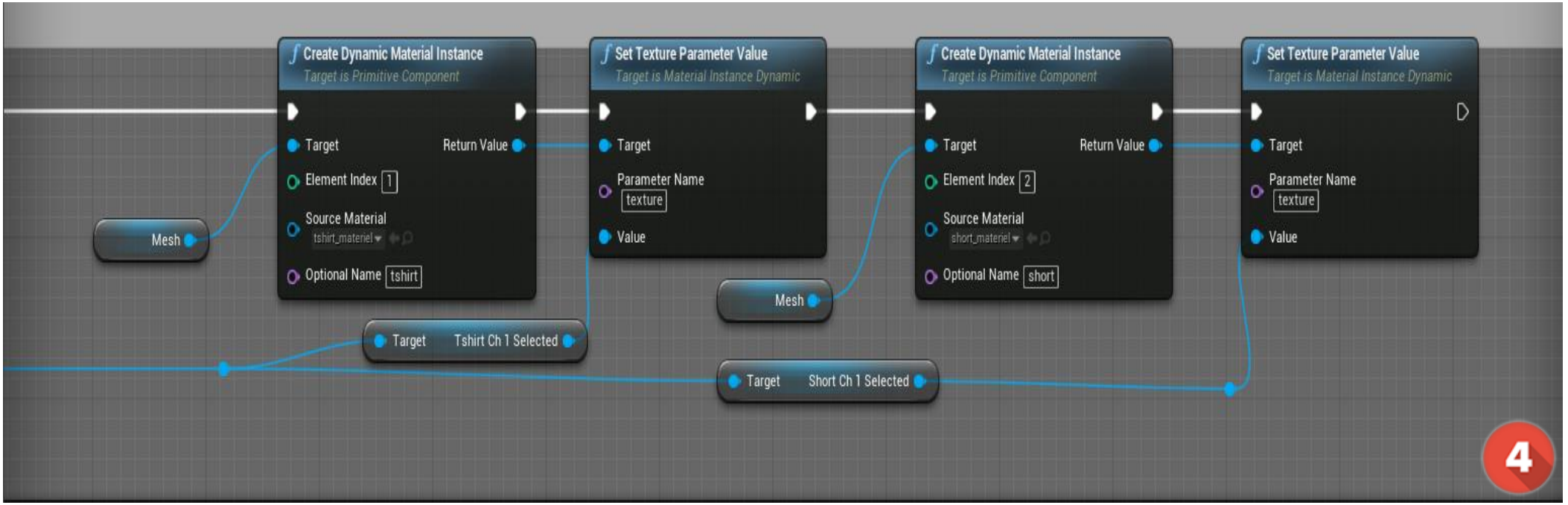

ATTACK HAMMER
ATTACK_EVENT
Custom Event
Branch
True
Condition
False
Mort?
Can Attacking
Branch
True
Condition
False
SET
Can Attacking
ACTOR TO CAMERA ORIENTATION BEGIN
Target is Hero
Target self
Play Anim Montage
Target is Character
Target self
Return Value
Anim Montage
In Play Rate 1,0
Start Section Name None
Attacks
Attack Count
GET
Soud Attack
GET
GetWorldLocation
Target is Scene Component
Target
Return Value
Capsule Component
valide
Custom Event
SET
Can Attacking
ResetCombo
Custom Event
SET
Attack Count 0
SET
Can Attacking
1


Play Sound at Location
Sound
Location
Rotation
X 0,0 Y 0,0 Z 0,0
Volume Multiplier 1,0
Pitch Multiplier 1,0
Start Time 0,0
Attenuation Settings
Concurrency Settings
Owning Actor
Make Noise
Target is Actor
Target self
Loudness 1,0
Noise Instigator
Noise Location
Max Range 0,0
Tag None
ACTOR TO CAMERA ORIENTATION END
Target is Hero
Target self
Attack Count
>=
LENGTH
Attacks
Branch
True
Condition
False
Attack Count
++
SET
Attack Count 0
2

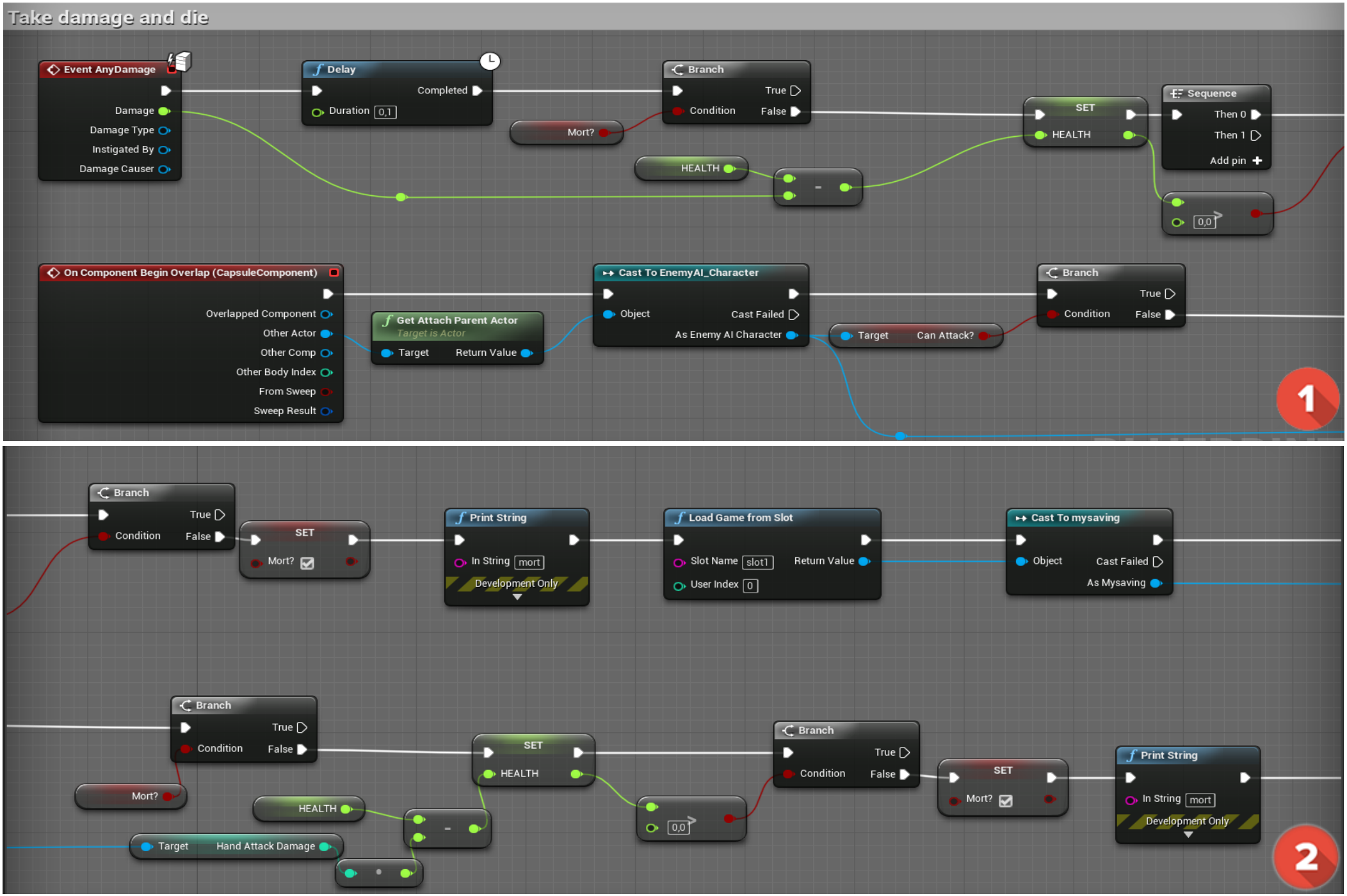
ANNEXE 3 : Blueprint Event BeginPlay du héros.

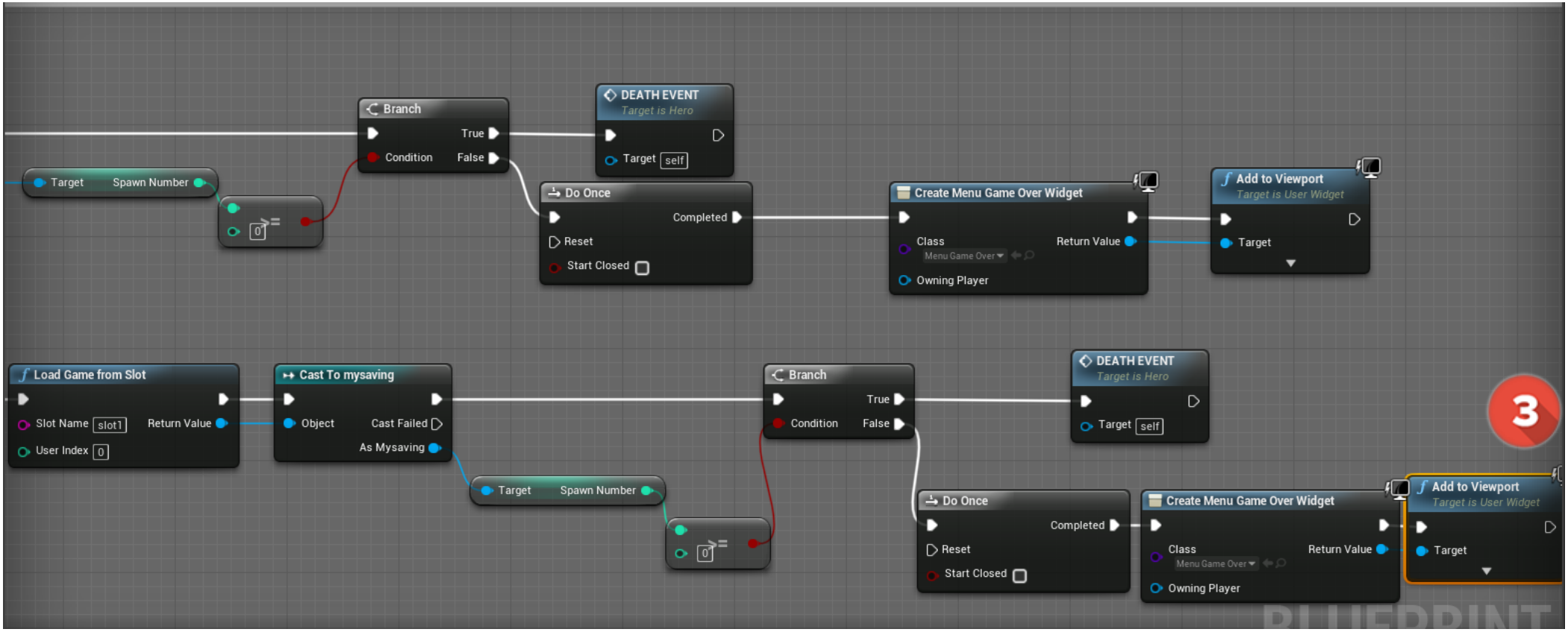

EVENT BEGIN AND PLAY
Event BeginPlay
SET
Show Mouse Cursor
Target
SET
View Pitch Min -40,0
Target
SET
View Pitch Max 20,0
Target
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
Get Player Controller
Player Index 0
Return Value
Get Player Camera Manager
Player Index 0
Return Value
1
SET
HEALTH
SpawnActor Bat Actor
Class Bat Actor
Return Value
Spawn Transform
Collision Handling Override
Default
Owner
Instigator
AttachToComponent
Target is Actor
Target
Parent
Socket Name weapon
Location Rule
Snap to Target
Rotation Rule
Snap to Target
Scale Rule
Snap to Target
Weld Simulated Bodies
Target HEALTH IN GAME
Get Socket Transform
Target is Scene Component
Target
Return Value
In Socket Name
weapon
Transform Space
RTS Parent Bone Space
Mesh
Mesh
2


Set Static Mesh
Target is Static Mesh Component
Target
Return Value
New Mesh
Cap
Target
Caps Ch 1 Selected
Leftshoes
Target
Leftshoes Ch 1 Selected
Rightshoes
Target
Rightshoes Ch 1 Selected
3

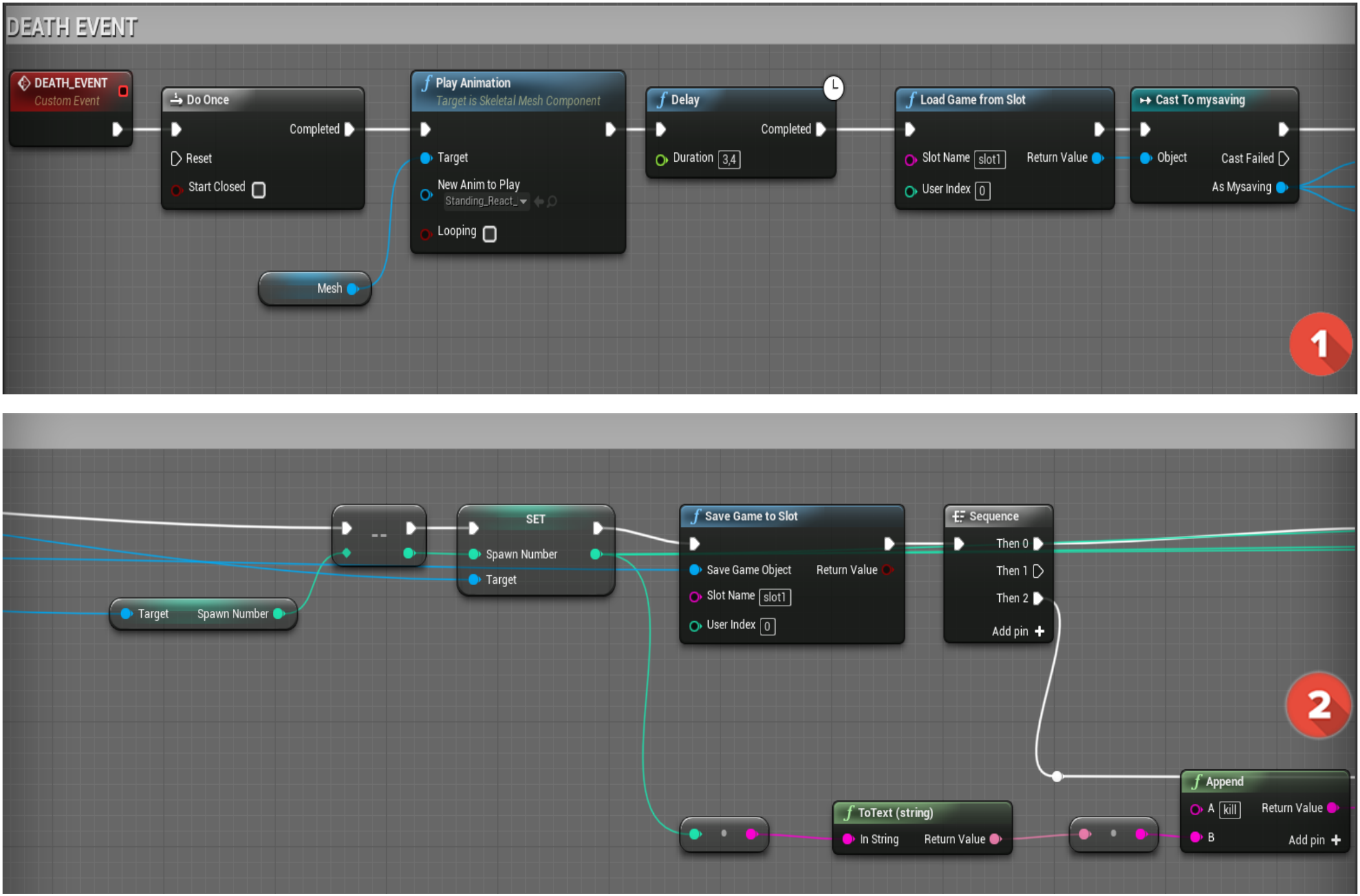

Create Dynamic Material Instance
Target is Primitive Component
Target
Return Value
Element Index 1
Source Material
tshirt_materiel
Optional Name tshirt
Mesh
Set Texture Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name
texture
Value
Target
Tshirt Ch 1 Selected
Create Dynamic Material Instance
Target is Primitive Component
Element Index 2
short_materiel
Optional Name short
Mesh
Target
Short Ch 1 Selected
4

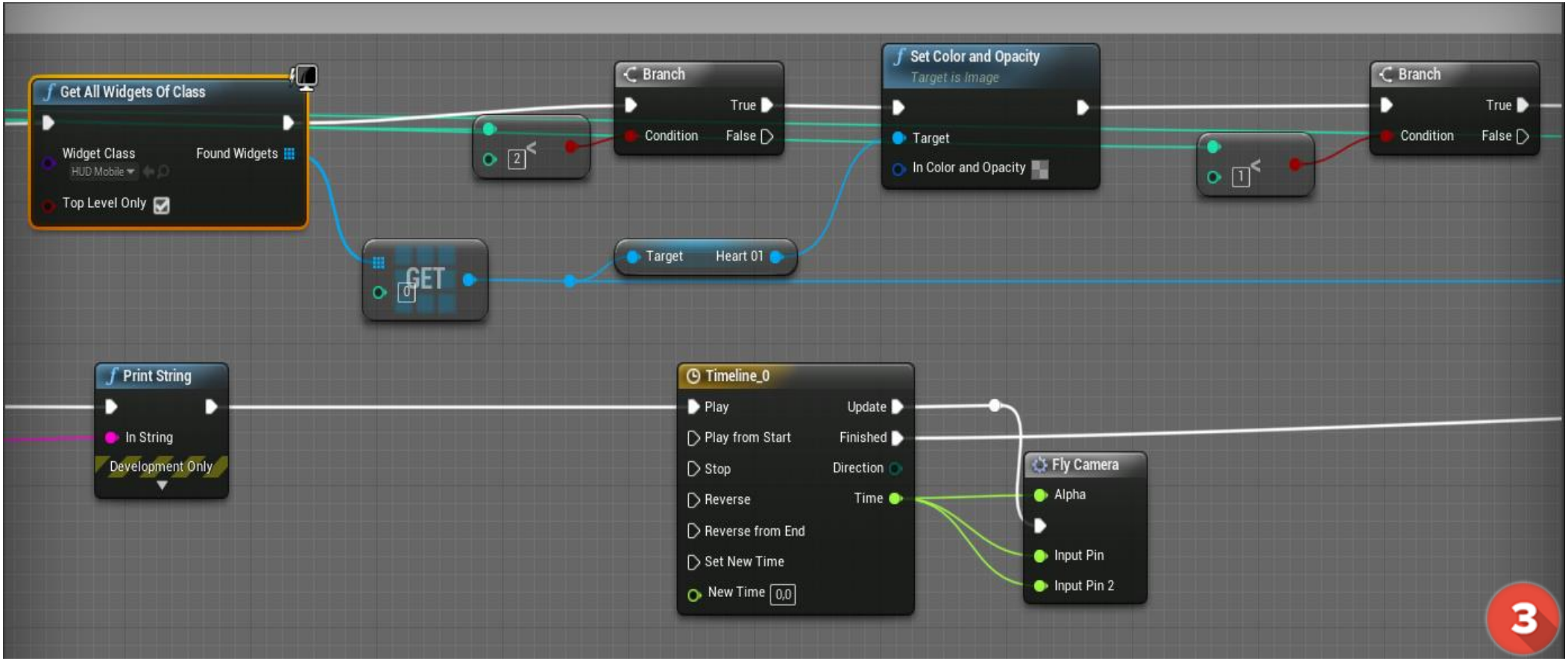
ANNEXE 4 : Blueprint Take damage And Die du héros.

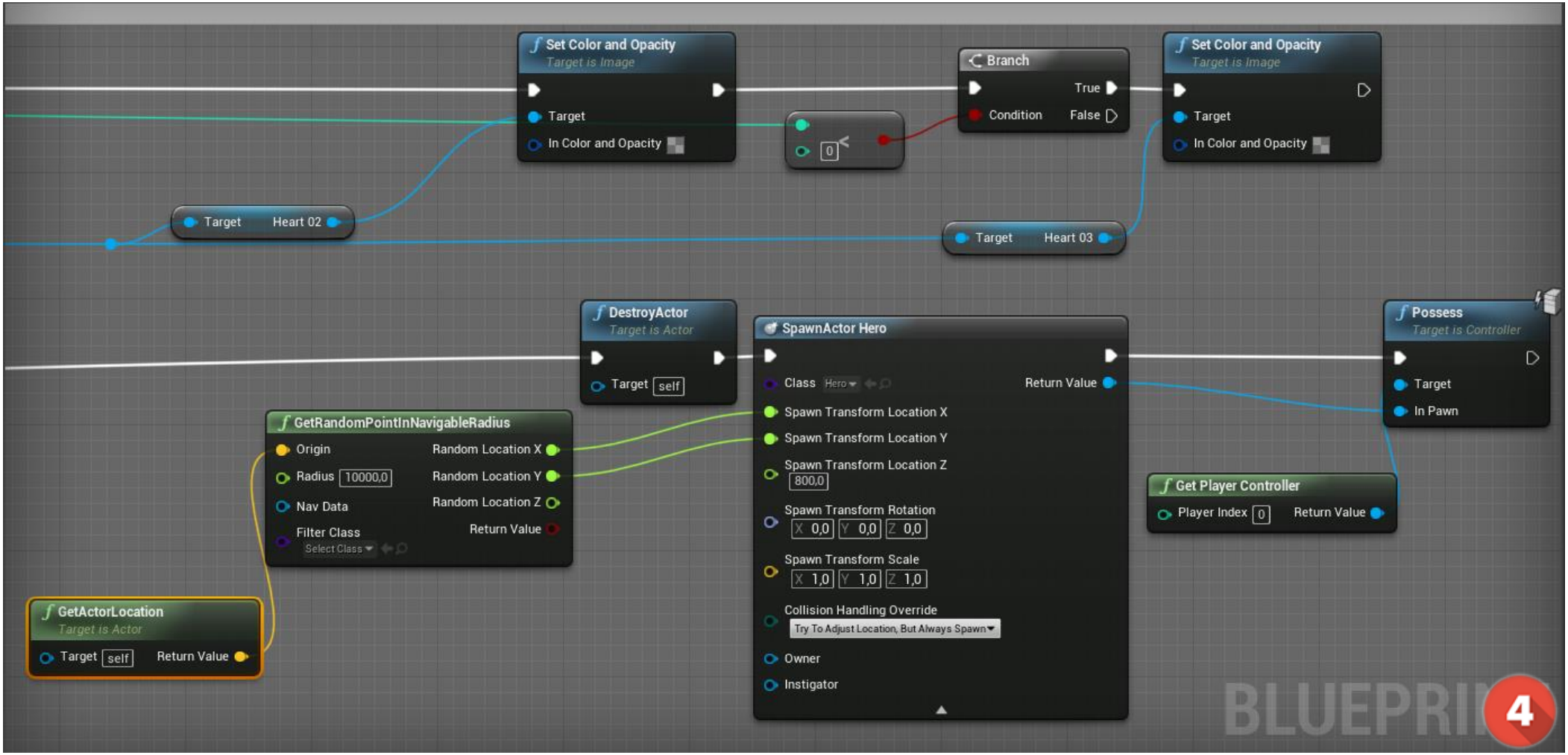

Take damage and die
Event AnyDamage
Damage
Damage Type
Instigated By
Damage Causer
Delay
Duration 0,1
Completed
Mort?
Branch
Condition
True
False
HEALTH
SET
HEALTH
Sequence
Then 0
Then 1
Add pin
0,0
On Component Begin Overlap (CapsuleComponent)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Get Attach Parent Actor
Target is Actor
Target
Return Value
Cast To EnemyAI_Character
Object
Cast Failed
As Enemy AI Character
Target
Can Attack?
Branch
Condition
True
False
1
Branch
Condition
True
False
SET
Mort?
Print String
In String mort
Development Only
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
Cast To mysaving
Object
Cast Failed
As Mysaving
Branch
Condition
True
False
Mort?
SET
HEALTH
HEALTH
Target
Hand Attack Damage
0,0
Branch
Condition
True
False
SET
Mort?
Print String
In String mort
Development Only
2


Branch
True
Condition
False
DEATH EVENT
Target is Hero
Target
self
Target
Spawn Number
0
Do Once
Completed
Reset
Start Closed
Create Menu Game Over Widget
Class
Menu Game Over
Return Value
Owning Player
Add to Viewport
Target is User Widget
Target
Load Game from Slot
Slot Name
slot1
Return Value
User Index
0
Cast To mysaving
Object
Cast Failed
As Mysaving
Branch
True
Condition
False
DEATH EVENT
Target is Hero
Target
self
3
Target
Spawn Number
0
Do Once
Completed
Reset
Start Closed
Create Menu Game Over Widget
Class
Menu Game Over
Return Value
Owning Player
Add to Viewport
Target is User Widget
Target

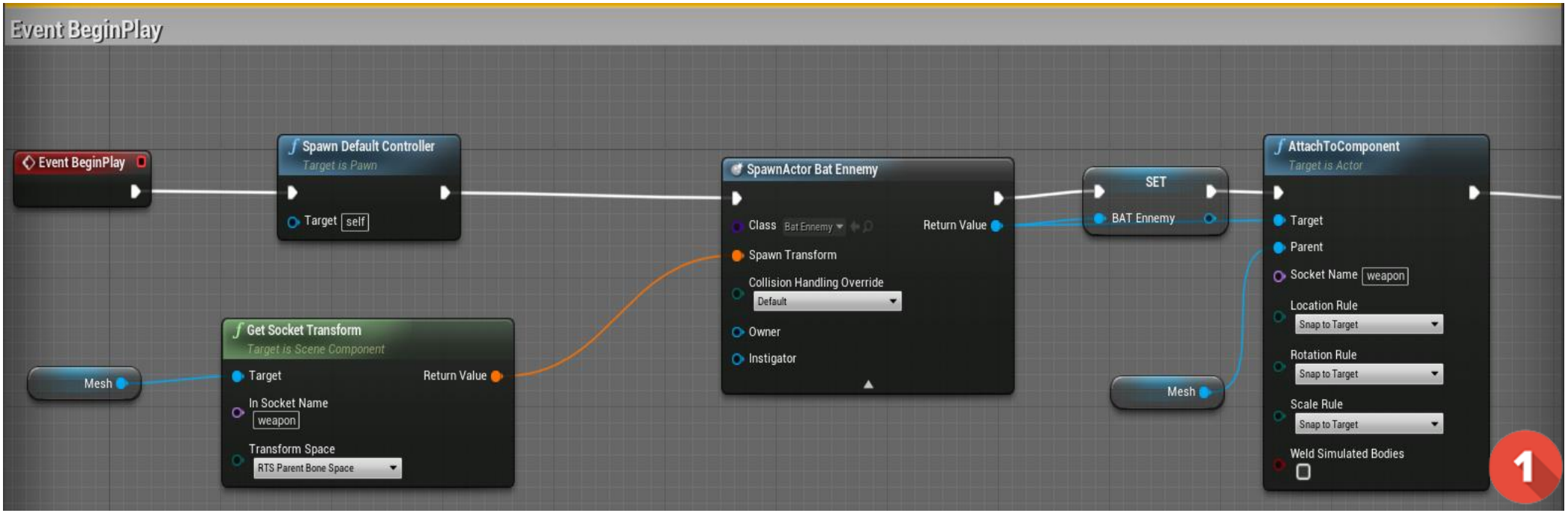
ANNEXE 5 : Blueprint Death Event du héros

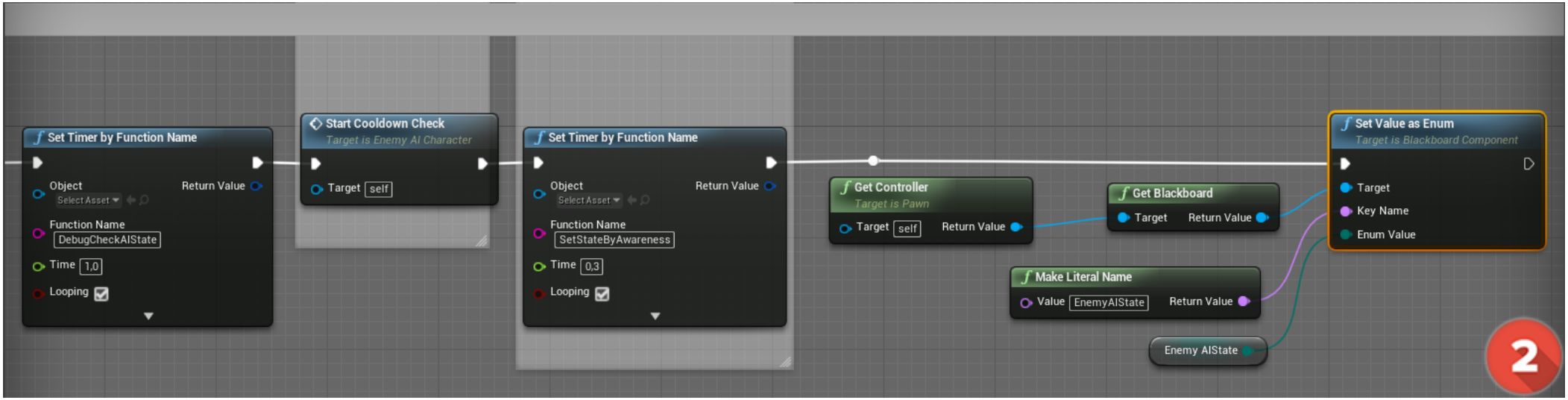

DEATH EVENT
DEATH_EVENT
Custom Event
Do Once
Completed
Reset
Start Closed
Play Animation
Target is Skeletal Mesh Component
Target
New Anim to Play
Standing_React_
Looping
Mesh
Delay
Completed
Duration 3,4
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
1
--
SET
Spawn Number
Target
Target
Spawn Number
Save Game to Slot
Save Game Object
Return Value
Slot Name slot1
User Index 0
Sequence
Then 0
Then 1
Then 2
Add pin
2
ToText (string)
In String
Return Value
Append
A kill
Return Value
B
Add pin


Get All Widgets Of Class
Widget Class
HUD Mobile
Found Widgets
Top Level Only
2
Branch
True
Condition
False
Set Color and Opacity
Target is Image
Target
In Color and Opacity
1
Branch
True
Condition
False
GET
0
Target
Heart 01
Print String
In String
Development Only
Timeline_0
Play
Play from Start
Stop
Reverse
Reverse from End
Set New Time
New Time
0,0
Update
Finished
Direction
Time
Fly Camera
Alpha
Input Pin
Input Pin 2
3

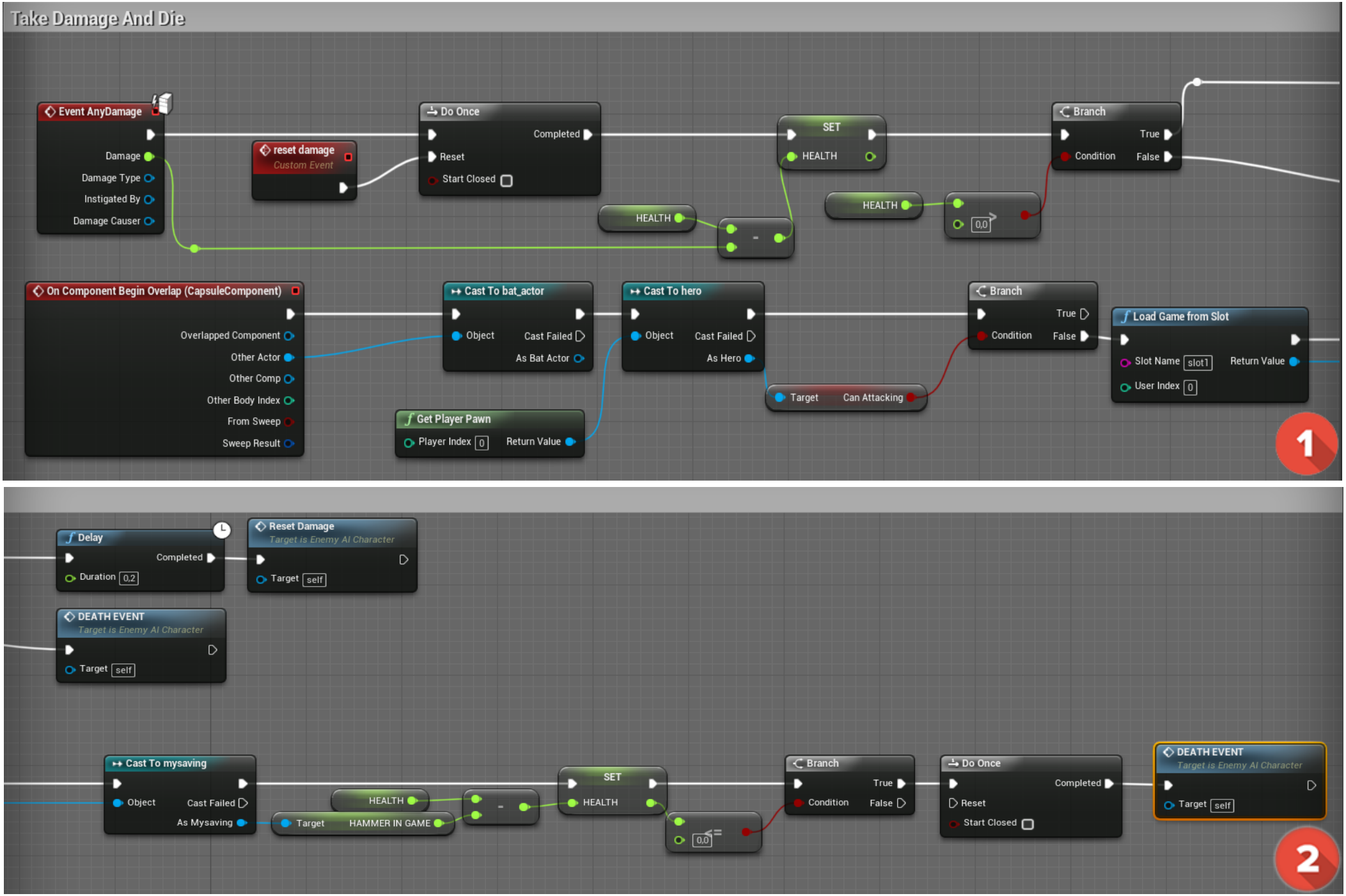

Set Color and Opacity
Target is Image
Target
In Color and Opacity
Branch
True
Condition
False
0
Set Color and Opacity
Target is Image
Target
In Color and Opacity
Target
Heart 02
Target
Heart 03
DestroyActor
Target is Actor
Target
self
SpawnActor Hero
Class
Hero
Return Value
Spawn Transform Location X
Spawn Transform Location Y
Spawn Transform Location Z
800,0
Spawn Transform Rotation
X 0,0
Y 0,0
Z 0,0
Spawn Transform Scale
X 1,0
Y 1,0
Z 1,0
Collision Handling Override
Try To Adjust Location, But Always Spawn
Owner
Instigator
Possess
Target is Controller
Target
In Pawn
GetRandomPointInNavigableRadius
Origin
Radius
10000,0
Nav Data
Filter Class
Select Class
Random Location X
Random Location Y
Random Location Z
Return Value
Get Player Controller
Player Index
0
Return Value
GetActorLocation
Target is Actor
Target
self
Return Value
BLUEPRI
4

ANNEXE 6 : Blueprint Event BeginPlay des ennemis.

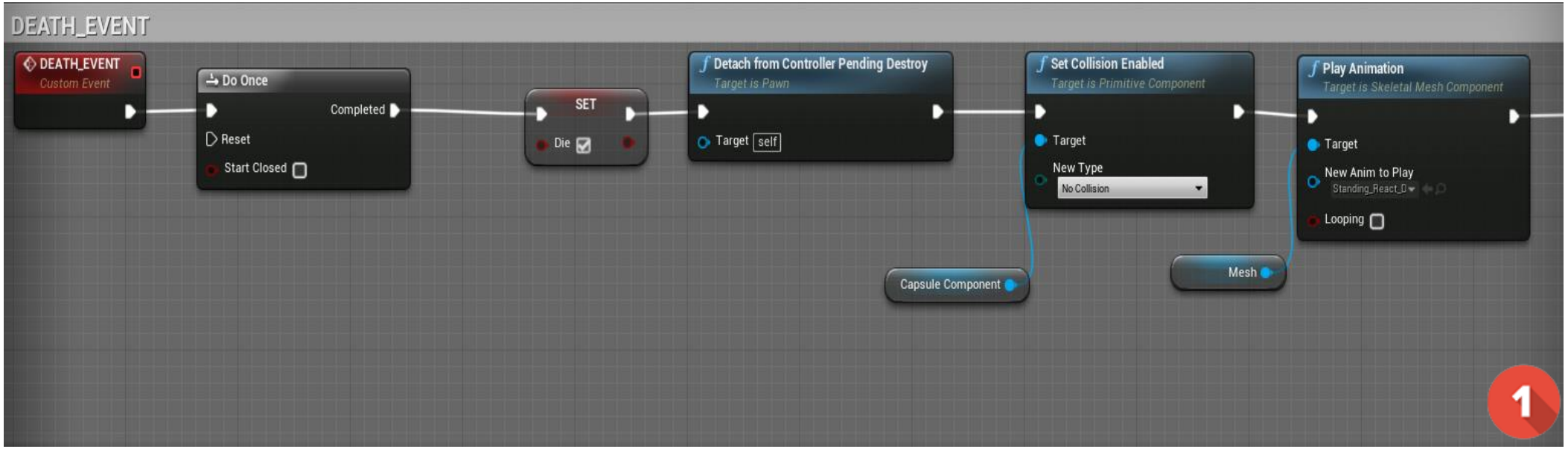

Event BeginPlay
Event BeginPlay
Spawn Default Controller
Target is Pawn
Target self
SpawnActor Bat Ennemy
Class Bat Ennemy
Return Value
Spawn Transform
Collision Handling Override
Default
Owner
Instigator
SET
BAT Ennemy
AttachToComponent
Target is Actor
Target
Parent
Socket Name weapon
Location Rule
Snap to Target
Rotation Rule
Snap to Target
Scale Rule
Snap to Target
Weld Simulated Bodies
Get Socket Transform
Target is Scene Component
Target
Return Value
In Socket Name
weapon
Transform Space
RTS Parent Bone Space
Mesh
Mesh
1

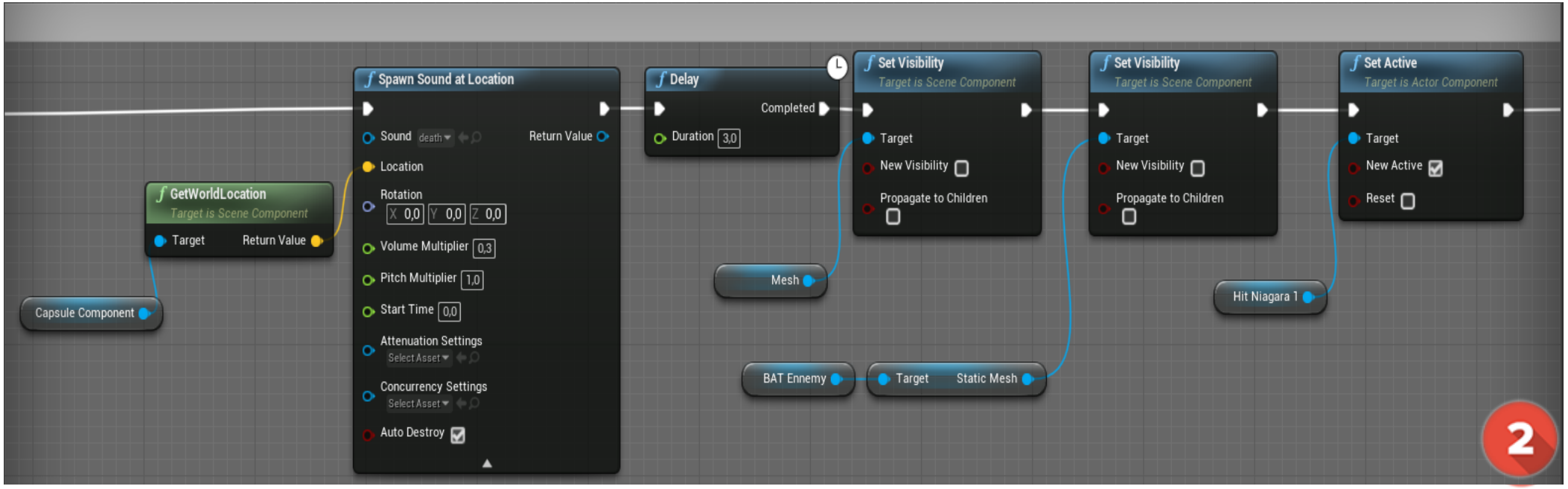

Set Timer by Function Name
Object
Return Value
Function Name
DebugCheckAIState
Time 1,0
Looping
Start Cooldown Check
Target is Enemy AI Character
Target self
Set Timer by Function Name
Object
Return Value
Function Name
SetStateByAwareness
Time 0,3
Looping
Get Controller
Target is Pawn
Target self
Return Value
Get Blackboard
Target
Return Value
Make Literal Name
Value EnemyAIState
Return Value
Enemy AIState
Set Value as Enum
Target is Blackboard Component
Target
Key Name
Enum Value
2

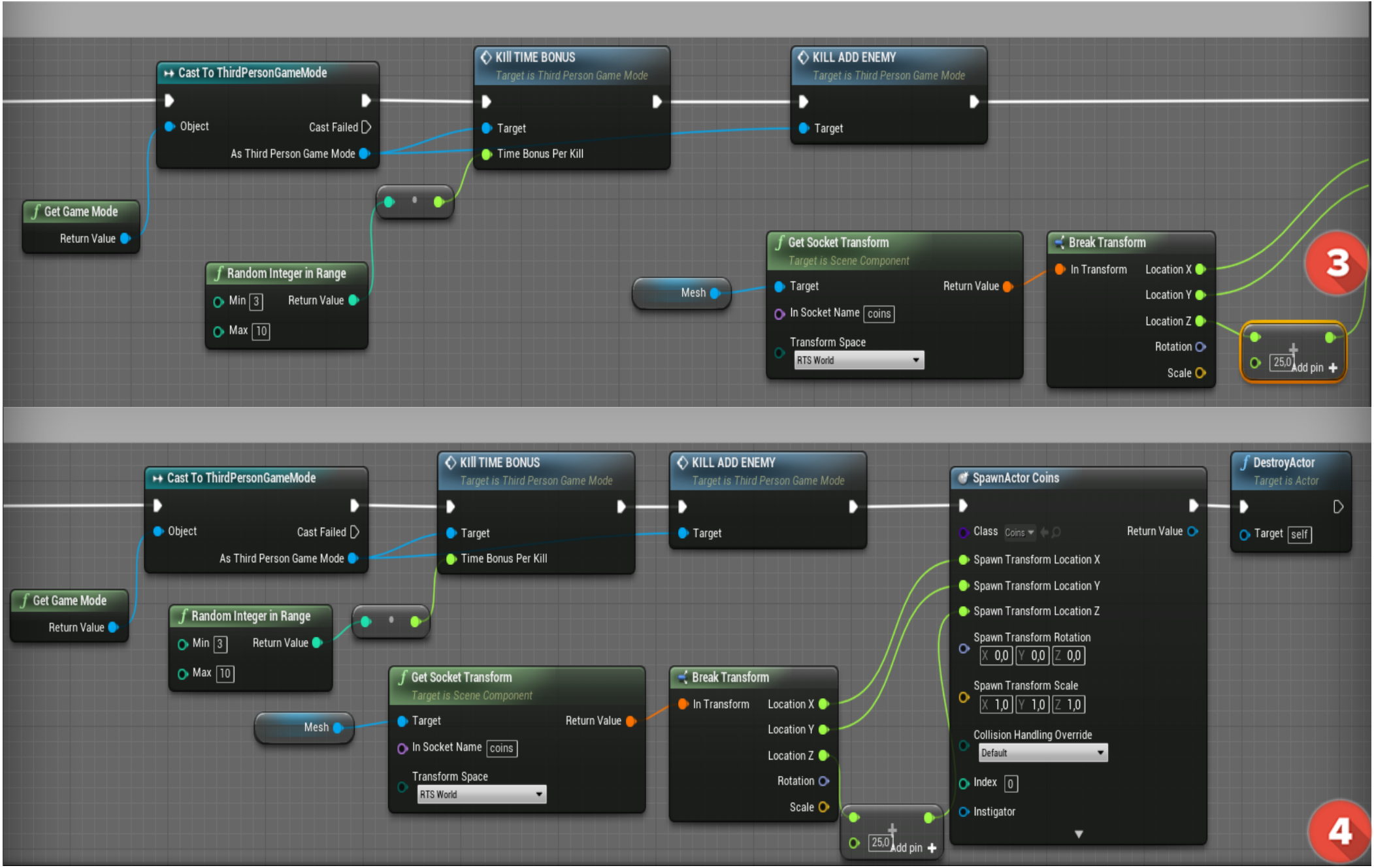
ANNEXE 7 : Blueprint Take damage And Die de l’ennemi.


Take Damage And Die
Event AnyDamage
Damage
Damage Type
Instigated By
Damage Causer
reset damage
Custom Event
Do Once
Reset
Start Closed
Completed
HEALTH
SET
HEALTH
Branch
Condition
True
False
0,0
On Component Begin Overlap (CapsuleComponent)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Cast To bat_actor
Object
Cast Failed
As Bat Actor
Cast To hero
As Hero
Get Player Pawn
Player Index
Return Value
Target
Can Attacking
Load Game from Slot
Slot Name
slot1
User Index
1
Delay
Duration
0,2
Reset Damage
Target is Enemy AI Character
Target
self
DEATH EVENT
Cast To mysaving
As Mysaving
HAMMER IN GAME
2

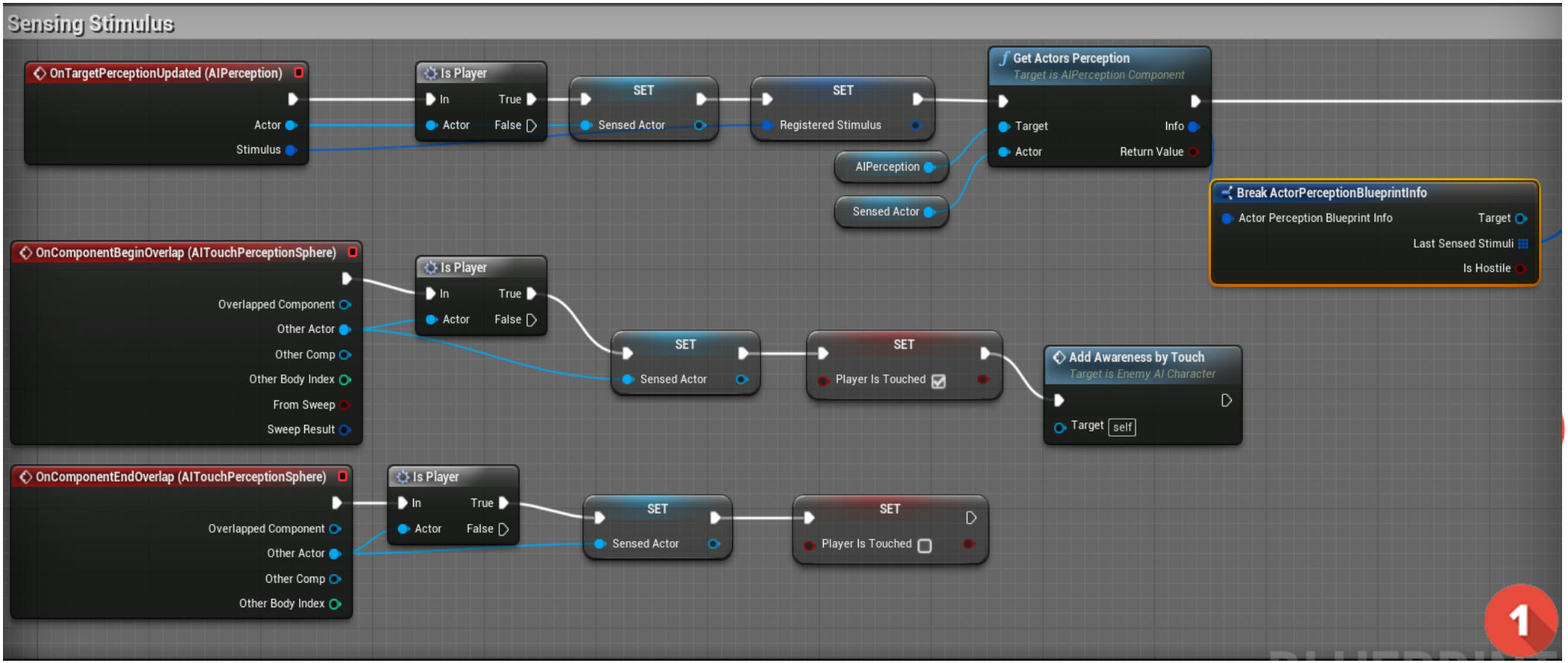
ANNEXE 8 : Blueprint Death Event de l’ennemi.

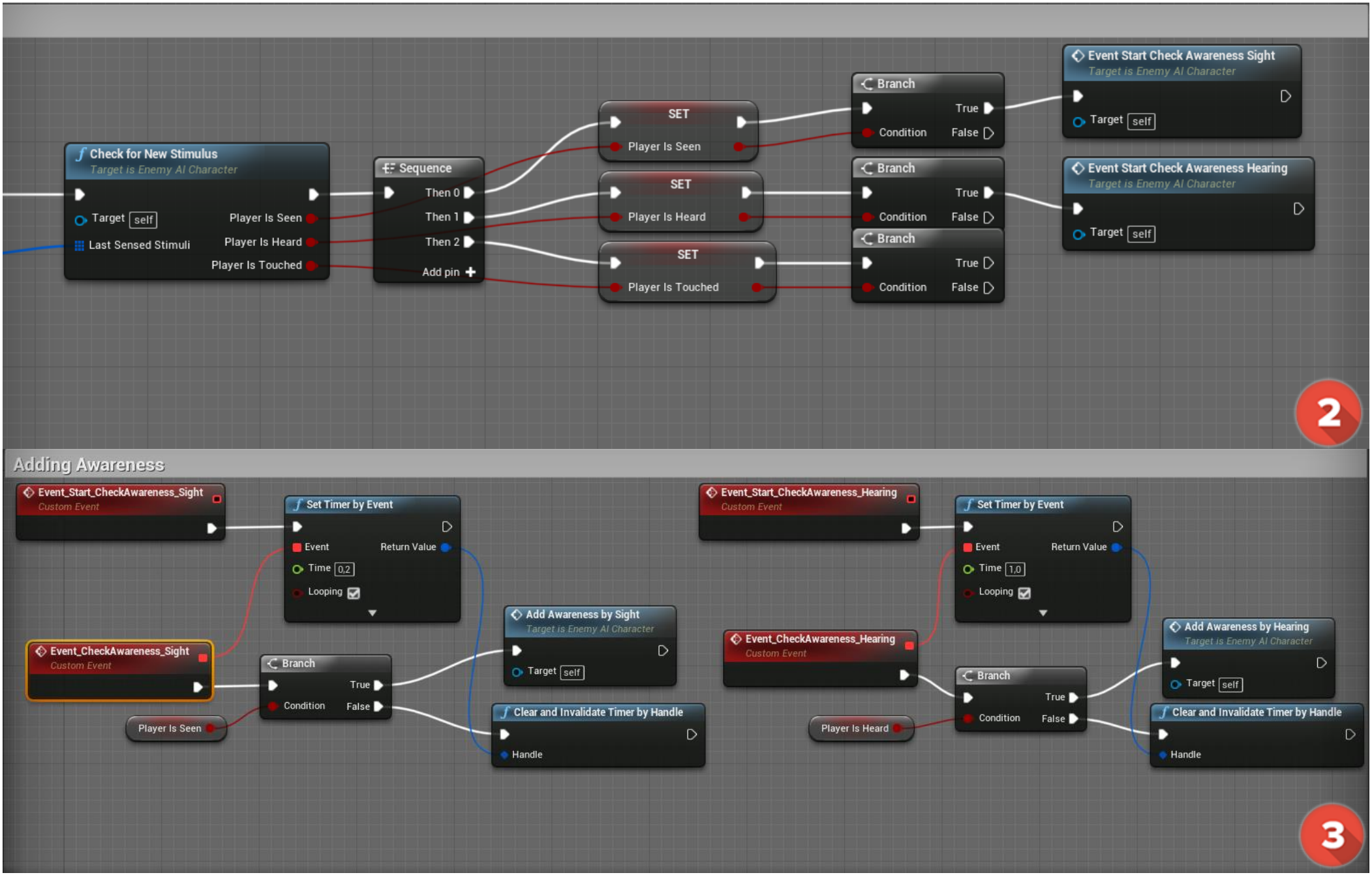

DEATH_EVENT
DEATH_EVENT
Custom Event
Do Once
Reset
Start Closed
Completed
SET
Die
Detach from Controller Pending Destroy
Target is Pawn
Target
self
Set Collision Enabled
Target is Primitive Component
Target
New Type
No Collision
Capsule Component
Play Animation
Target is Skeletal Mesh Component
Target
New Anim to Play
Standing_React_0
Looping
Mesh
1


Spawn Sound at Location
Sound
death
Location
Rotation
X 0,0
Y 0,0
Z 0,0
Volume Multiplier
0,3
Pitch Multiplier
1,0
Start Time
0,0
Attenuation Settings
Select Asset
Concurrency Settings
Select Asset
Auto Destroy
Return Value
GetWorldLocation
Target is Scene Component
Target
Return Value
Capsule Component
Delay
Duration
3,0
Completed
Mesh
Set Visibility
Target is Scene Component
Target
New Visibility
Propagate to Children
Set Visibility
Target is Scene Component
Target
New Visibility
Propagate to Children
BAT Ennemy
Target
Static Mesh
Set Active
Target is Actor Component
Target
New Active
Reset
Hit Niagara 1
2

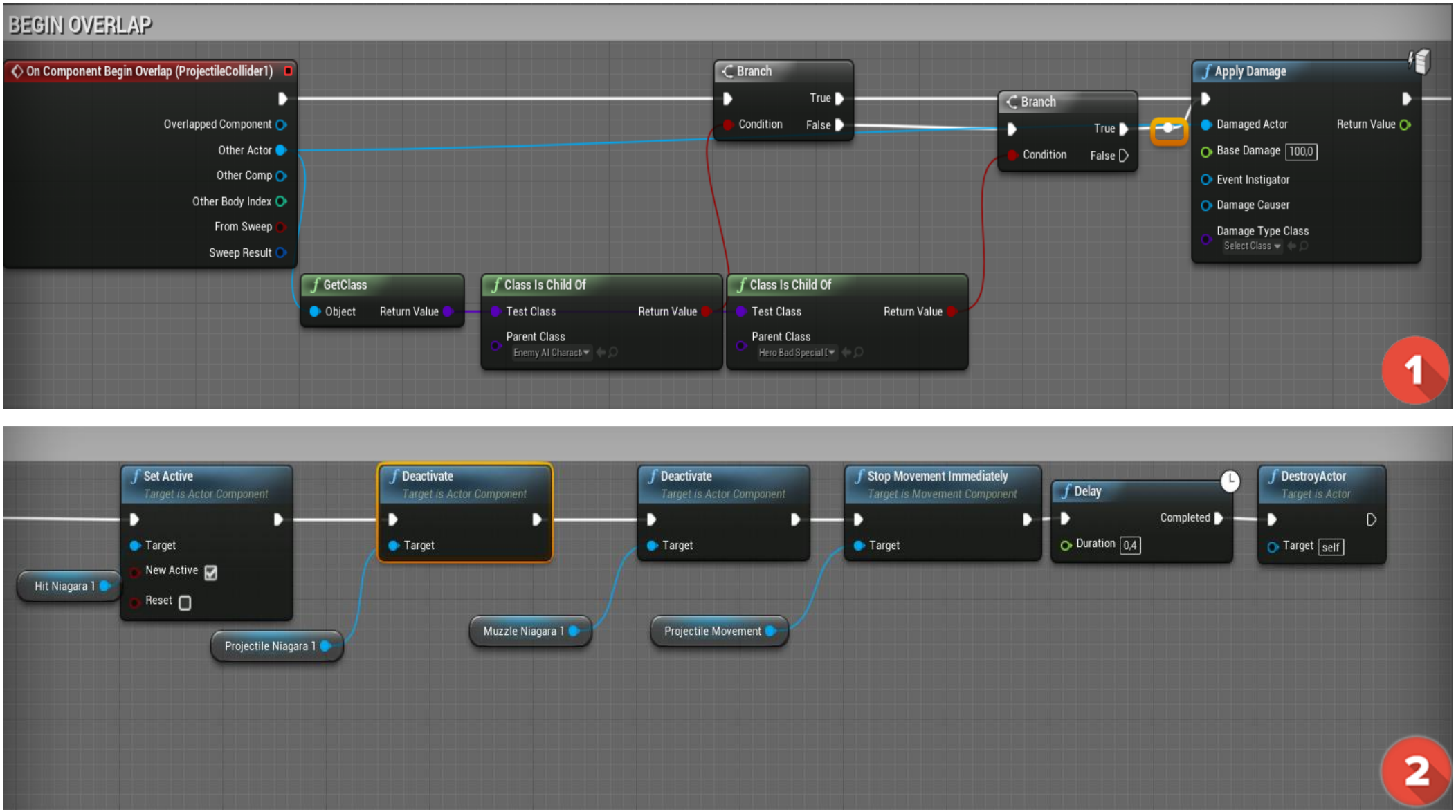

Cast To ThirdPersonGameMode
Object
Cast Failed
As Third Person Game Mode
Kill TIME BONUS
Target is Third Person Game Mode
Target
Time Bonus Per Kill
KILL ADD ENEMY
Target is Third Person Game Mode
Target
Get Game Mode
Return Value
Random Integer in Range
Min 3
Max 10
Return Value
Mesh
Get Socket Transform
Target is Scene Component
Target
In Socket Name coins
Transform Space
RTS World
Return Value
Break Transform
In Transform
Location X
Location Y
Location Z
Rotation
Scale
25,0
Add pin
3
Cast To ThirdPersonGameMode
Object
Cast Failed
As Third Person Game Mode
Kill TIME BONUS
Target is Third Person Game Mode
Target
Time Bonus Per Kill
KILL ADD ENEMY
Target is Third Person Game Mode
Target
SpawnActor Coins
Class Coins
Return Value
Spawn Transform Location X
Spawn Transform Location Y
Spawn Transform Location Z
Spawn Transform Rotation
X 0,0 Y 0,0 Z 0,0
Spawn Transform Scale
X 1,0 Y 1,0 Z 1,0
Collision Handling Override
Default
Index 0
Instigator
DestroyActor
Target is Actor
Target self
Get Game Mode
Return Value
Random Integer in Range
Min 3
Max 10
Return Value
Mesh
Get Socket Transform
Target is Scene Component
Target
In Socket Name coins
Transform Space
RTS World
Return Value
Break Transform
In Transform
Location X
Location Y
Location Z
Rotation
Scale
25,0
Add pin
4

ANNEXE 9 : Blueprint Sensing Stimulus de l’ennemi.

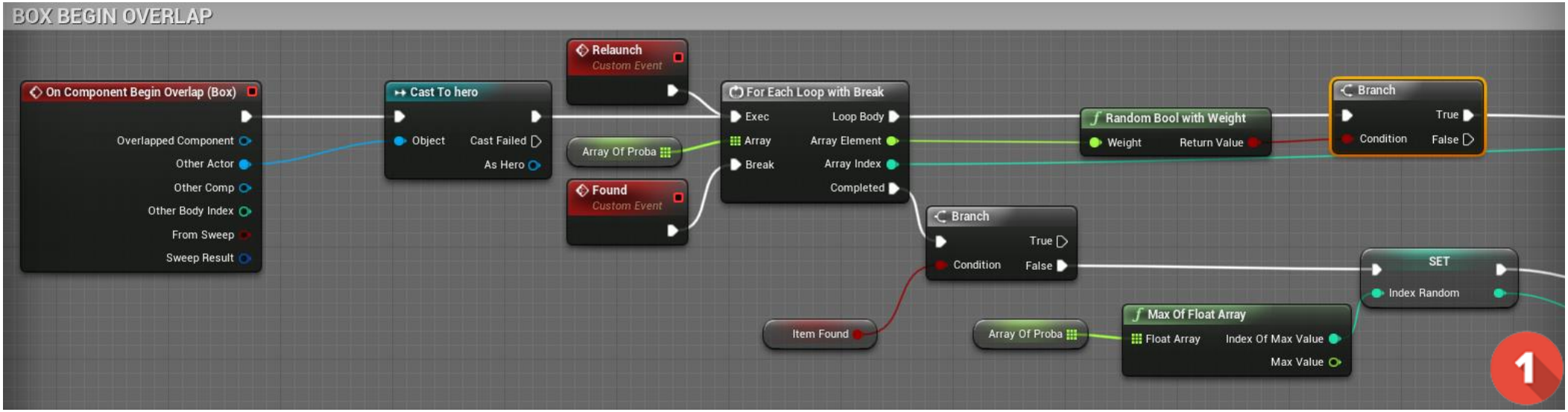

Sensing Stimulus
OnTargetPerceptionUpdated (AIPerception)
Actor
Stimulus
Is Player
In
True
Actor
False
SET
Sensed Actor
SET
Registered Stimulus
Get Actors Perception
Target is AIPerception Component
Target
Info
Actor
Return Value
AIPerception
Sensed Actor
Break ActorPerceptionBlueprintInfo
Actor Perception Blueprint Info
Target
Last Sensed Stimuli
Is Hostile
OnComponentBeginOverlap (AITouchPerceptionSphere)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Is Player
In
True
Actor
False
SET
Sensed Actor
SET
Player Is Touched
Add Awareness by Touch
Target is Enemy AI Character
Target
self
OnComponentEndOverlap (AITouchPerceptionSphere)
Overlapped Component
Other Actor
Other Comp
Other Body Index
Is Player
In
True
Actor
False
SET
Sensed Actor
SET
Player Is Touched
1

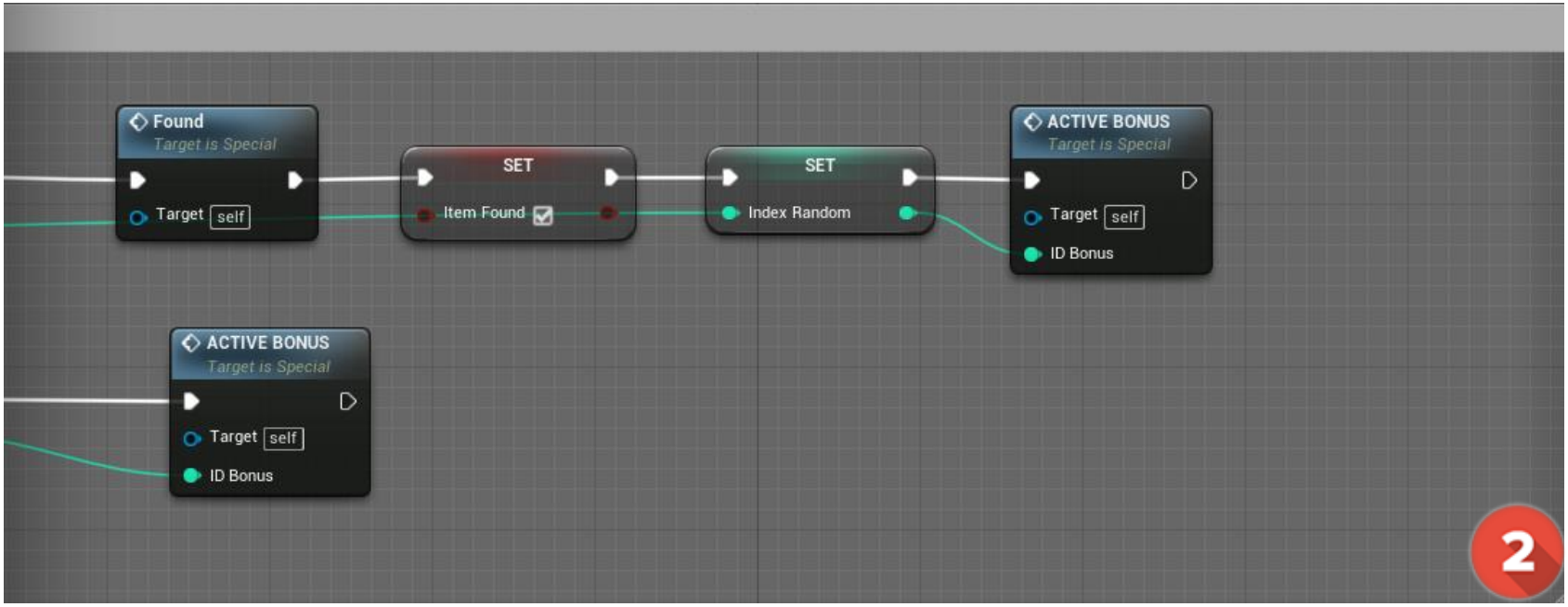

Check for New Stimulus
Target is Enemy AI Character
Target self
Last Sensed Stimuli
Player Is Seen
Player Is Heard
Player Is Touched
Sequence
Then 0
Then 1
Then 2
Add pin
SET
Player Is Seen
SET
Player Is Heard
SET
Player Is Touched
Branch
True
Condition
False
Branch
True
Condition
False
Branch
True
Condition
False
Event Start Check Awareness Sight
Target is Enemy AI Character
Target self
Event Start Check Awareness Hearing
Target is Enemy AI Character
Target self
2
Adding Awareness
Event_Start_CheckAwareness_Sight
Custom Event
Set Timer by Event
Event
Return Value
Time 0,2
Looping
Add Awareness by Sight
Target is Enemy AI Character
Target self
Event_CheckAwareness_Sight
Custom Event
Branch
True
Condition
False
Player Is Seen
Clear and Invalidate Timer by Handle
Handle
Event_Start_CheckAwareness_Hearing
Custom Event
Set Timer by Event
Event
Return Value
Time 1,0
Looping
Event_CheckAwareness_Hearing
Custom Event
Branch
True
Condition
False
Player Is Heard
Add Awareness by Hearing
Target is Enemy AI Character
Target self
Clear and Invalidate Timer by Handle
Handle
3

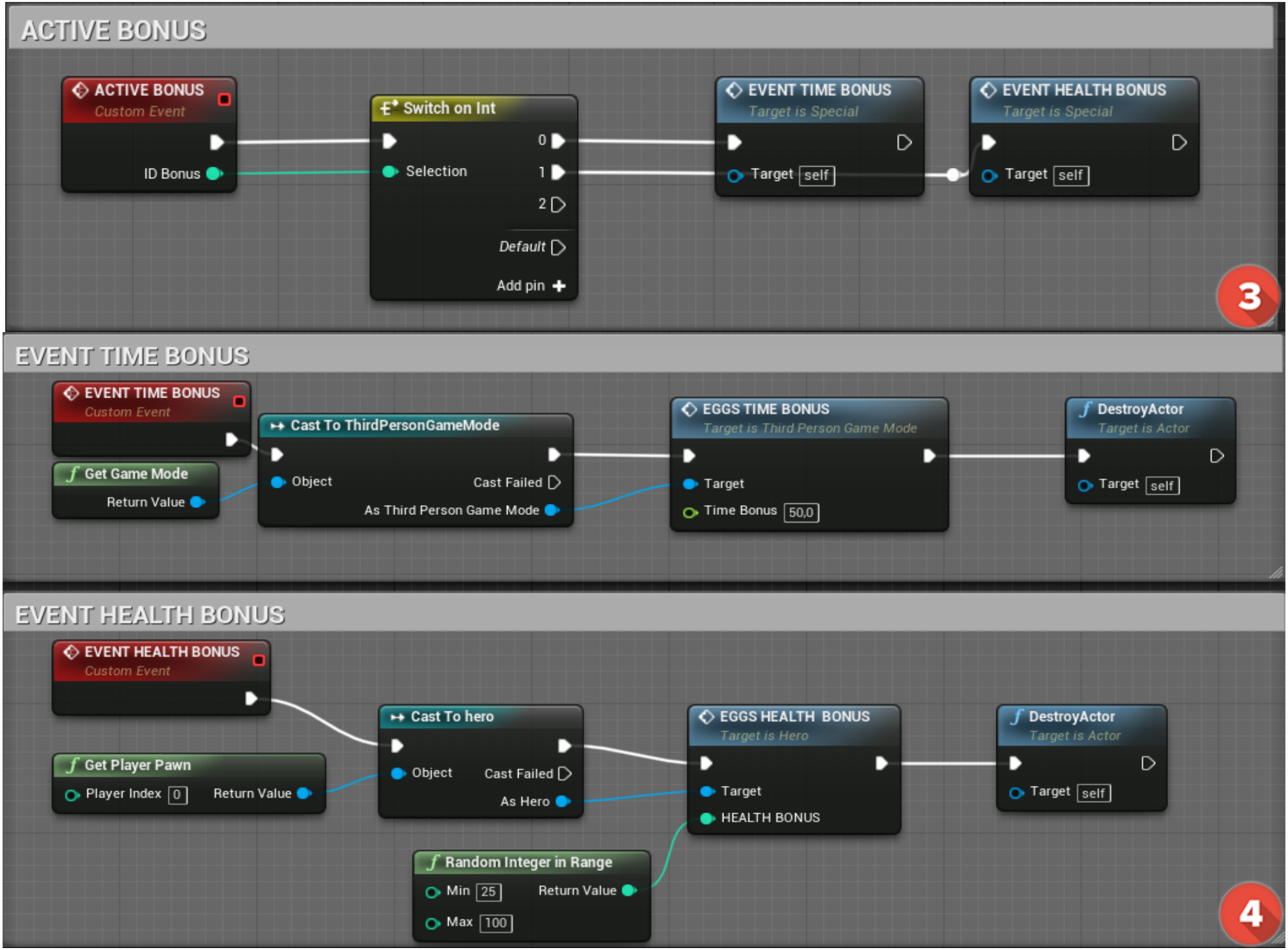
ANNEXE 10 : Blueprint Begin Overlap d’une bombe.


BEGIN OVERLAP
On Component Begin Overlap (ProjectileCollider1)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Branch
True
Condition
False
Branch
True
Condition
False
Apply Damage
Damaged Actor
Return Value
Base Damage
100,0
Event Instigator
Damage Causer
Damage Type Class
Select Class
GetClass
Object
Return Value
Class Is Child Of
Test Class
Return Value
Parent Class
Enemy AI Charact
Class Is Child Of
Test Class
Return Value
Parent Class
Hero Bad Special I
1
Set Active
Target is Actor Component
Target
New Active
Reset
Hit Niagara 1
Projectile Niagara 1
Deactivate
Target is Actor Component
Target
Muzzle Niagara 1
Deactivate
Target is Actor Component
Target
Projectile Movement
Stop Movement Immediately
Target is Movement Component
Target
Delay
Completed
Duration
0,4
DestroyActor
Target is Actor
Target
self
2

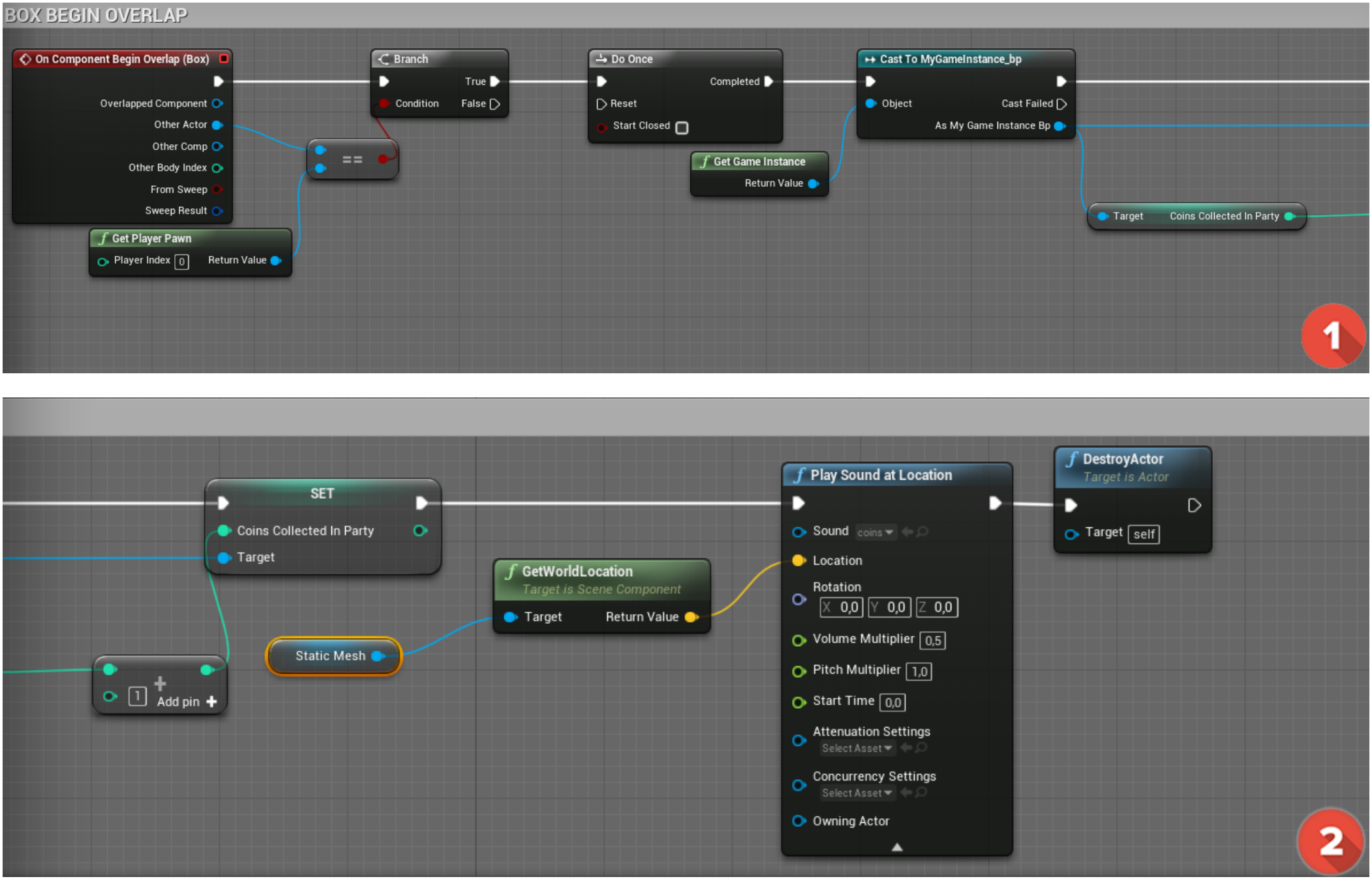
ANNEXE 11 : Blueprint Begin Overlap des œufs de bonus.


BOX BEGIN OVERLAP
On Component Begin Overlap (Box)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Cast To hero
Object
Cast Failed
As Hero
Relaunch
Custom Event
Array Of Proba
Found
Custom Event
For Each Loop with Break
Exec
Array
Break
Loop Body
Array Element
Array Index
Completed
Random Bool with Weight
Weight
Return Value
Branch
Condition
True
False
Branch
Condition
True
False
Item Found
Array Of Proba
Max Of Float Array
Float Array
Index Of Max Value
Max Value
SET
Index Random
1

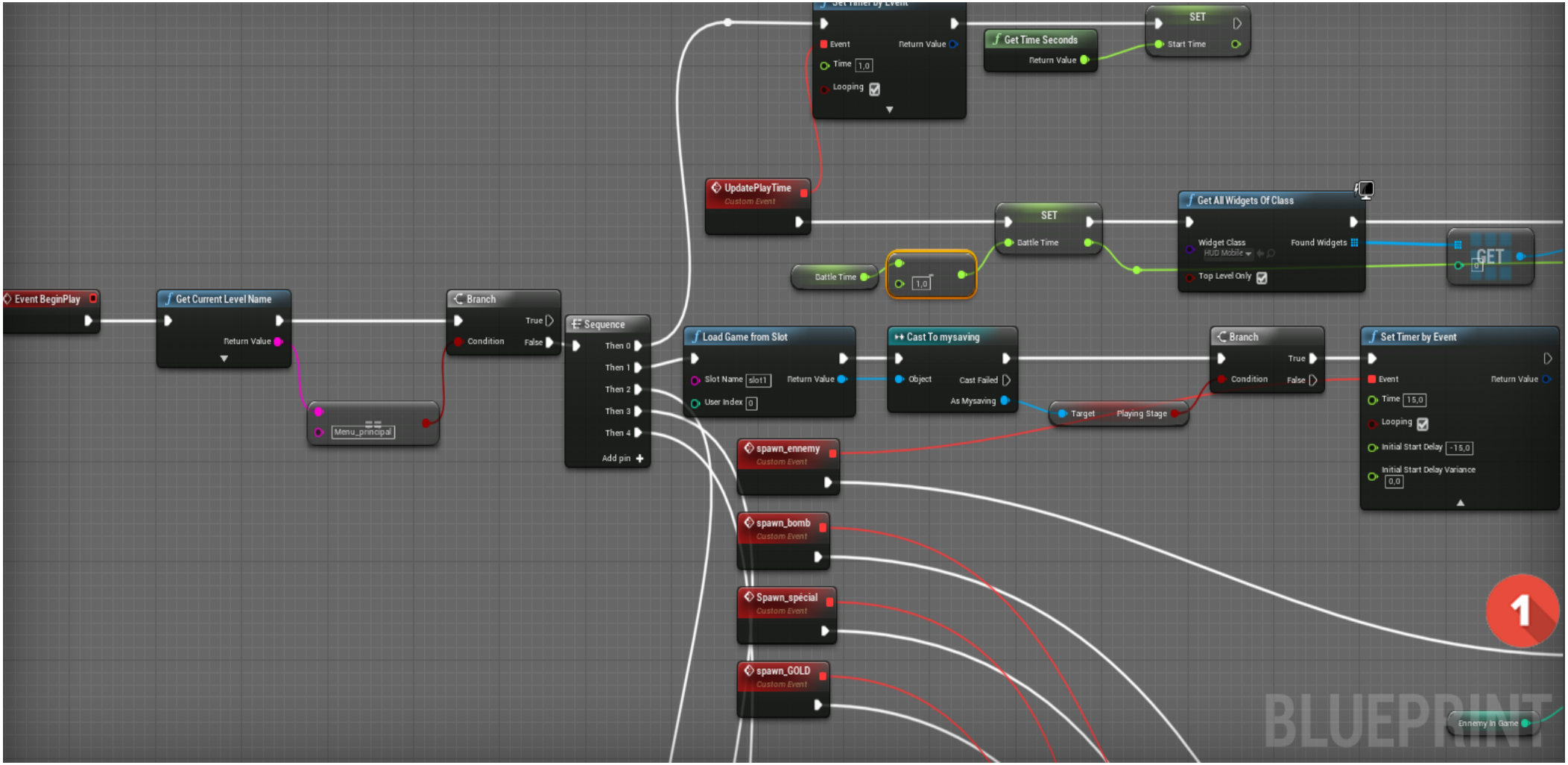

Found
Target is Special
Target
self
SET
Item Found
SET
Index Random
ACTIVE BONUS
Target is Special
Target
self
ID Bonus
ACTIVE BONUS
Target is Special
Target
self
ID Bonus
2


ACTIVE BONUS
ACTIVE BONUS
Custom Event
ID Bonus
Switch on Int
Selection
0
1
2
Default
Add pin
EVENT TIME BONUS
Target is Special
Target self
EVENT HEALTH BONUS
Target is Special
Target self
3
EVENT TIME BONUS
EVENT TIME BONUS
Custom Event
Get Game Mode
Return Value
Cast To ThirdPersonGameMode
Object
Cast Failed
As Third Person Game Mode
EGGS TIME BONUS
Target is Third Person Game Mode
Target
Time Bonus 50,0
DestroyActor
Target is Actor
Target self
EVENT HEALTH BONUS
EVENT HEALTH BONUS
Custom Event
Get Player Pawn
Player Index 0
Return Value
Cast To hero
Object
Cast Failed
As Hero
EGGS HEALTH BONUS
Target is Hero
Target
HEALTH BONUS
DestroyActor
Target is Actor
Target self
Random Integer in Range
Min 25
Max 100
Return Value
4

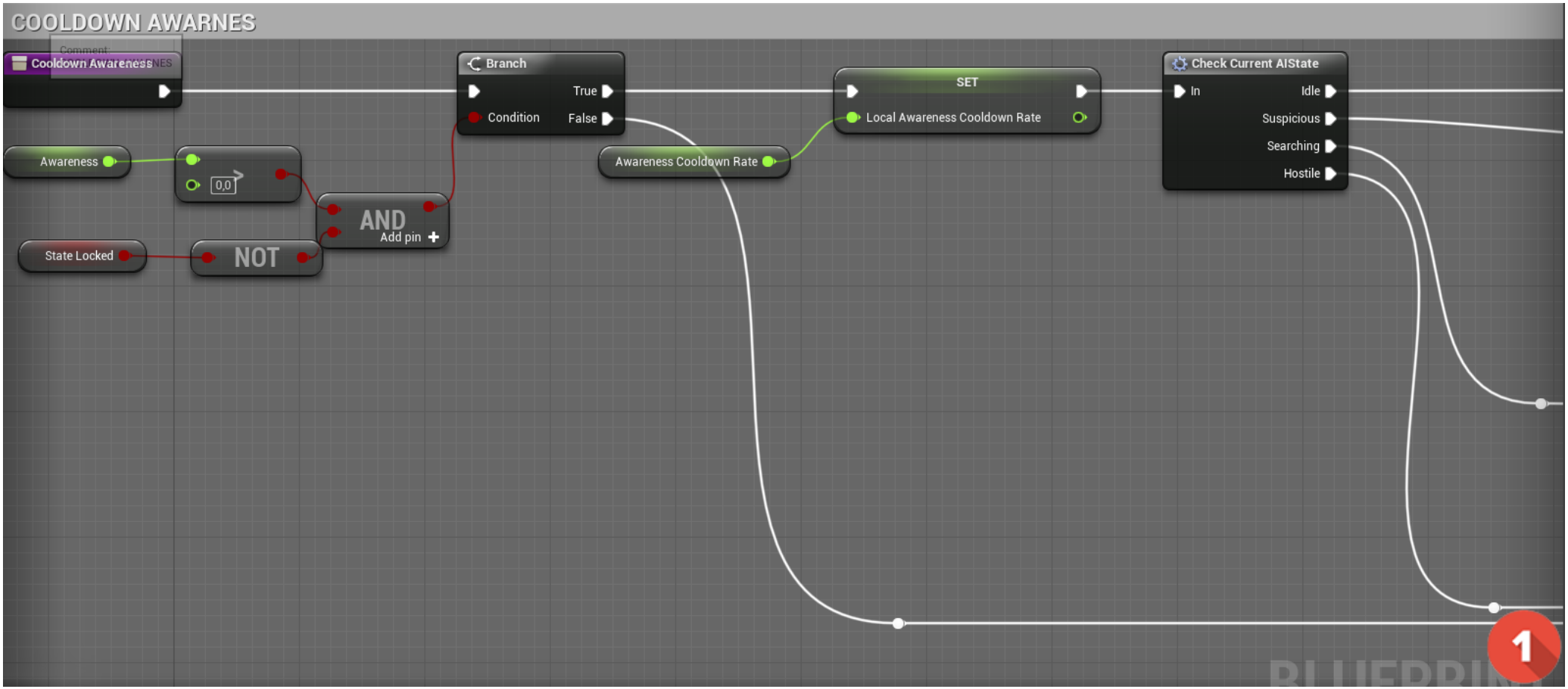
ANNEXE 12 : Blueprint Begin Overlap des pièces.

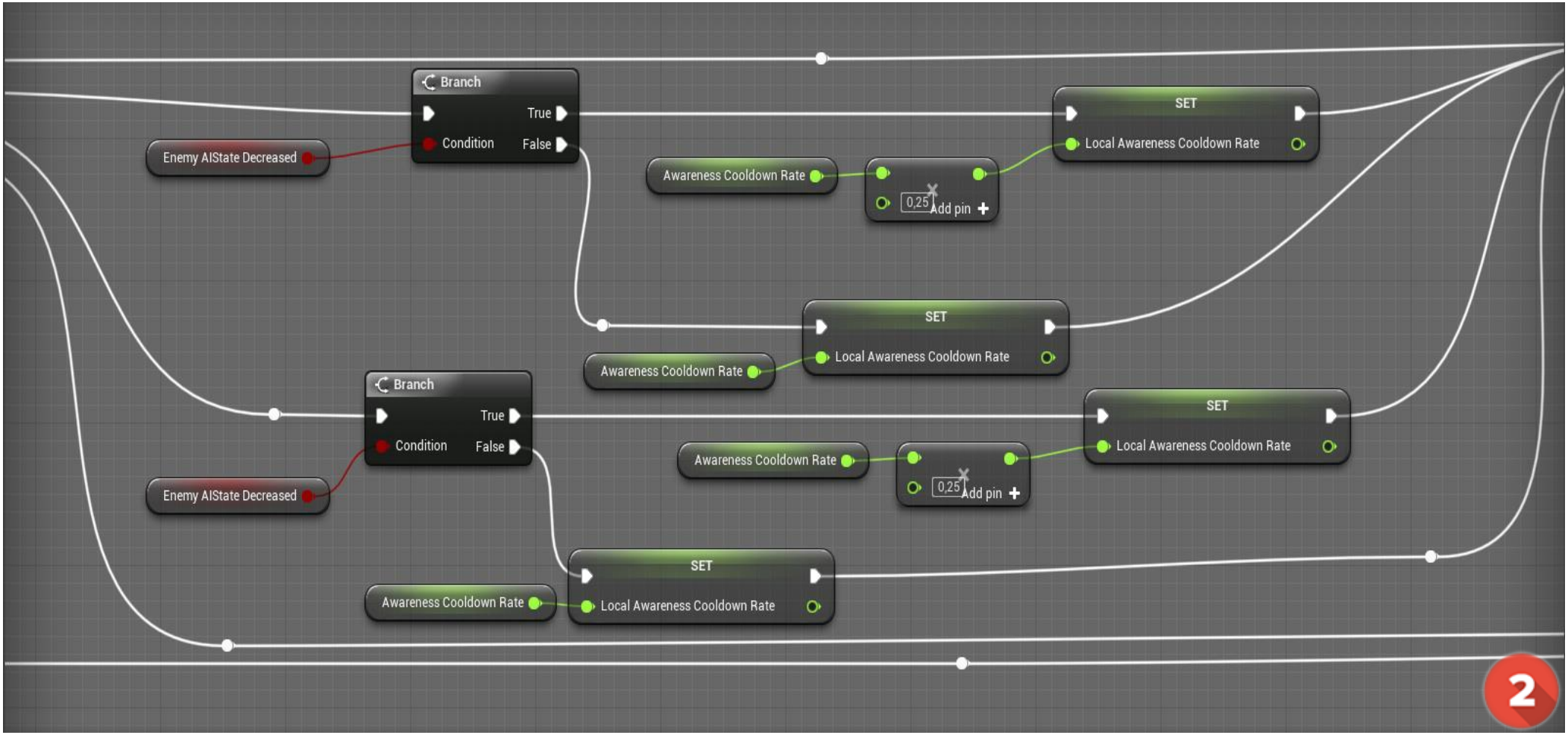

BOX BEGIN OVERLAP
On Component Begin Overlap (Box)
Overlapped Component
Other Actor
Other Comp
Other Body Index
From Sweep
Sweep Result
Branch
True
Condition
False
Do Once
Completed
Reset
Start Closed
Cast To MyGameInstance_bp
Object
Cast Failed
As My Game Instance Bp
Get Game Instance
Return Value
Get Player Pawn
Player Index 0
Return Value
Target
Coins Collected In Party
1
SET
Coins Collected In Party
Target
GetWorldLocation
Target is Scene Component
Target
Return Value
Static Mesh
Add pin
Play Sound at Location
Sound coins
Location
Rotation
X 0,0 Y 0,0 Z 0,0
Volume Multiplier 0,5
Pitch Multiplier 1,0
Start Time 0,0
Attenuation Settings
Select Asset
Concurrency Settings
Select Asset
Owning Actor
DestroyActor
Target is Actor
Target self
2

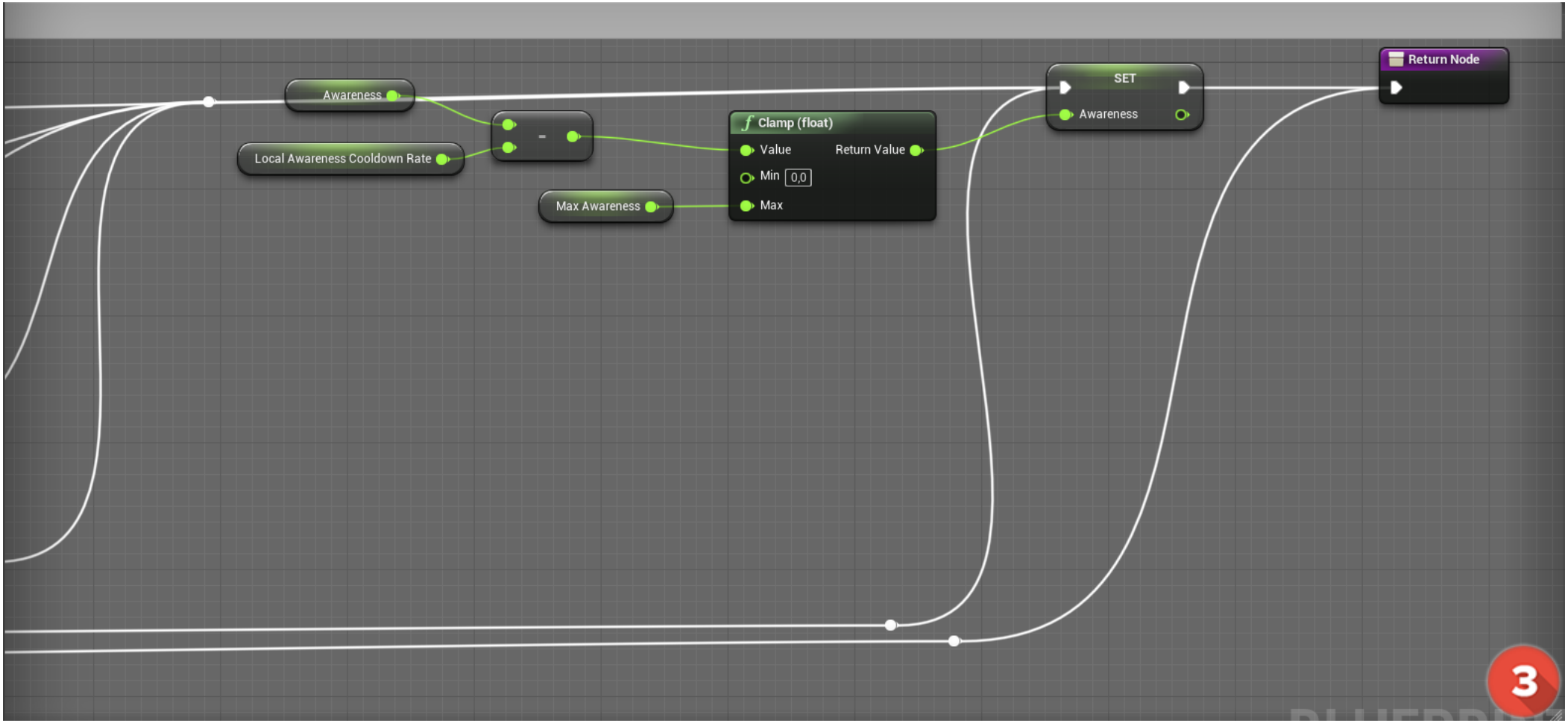
ANNEXE 13 : Partie du blueprint Game Mode.


Event BeginPlay
Get Current Level Name
Return Value
Menu_principal
Branch
Condition
True
False
Sequence
Then 0
Then 1
Then 2
Then 3
Then 4
Add pin
Event
Return Value
Time 1,0
Looping
Get Time Seconds
Return Value
SET
Start Time
UpdatePlayTime
Custom Event
SET
Battle Time
Battle Time
1,0
Get All Widgets Of Class
Widget Class
HUD Mobile
Found Widgets
Top Level Only
GET
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
Cast To mysaving
Object
Cast Failed
As Mysaving
Target
Playing Stage
Branch
Condition
True
False
Set Timer by Event
Event
Return Value
Time 15,0
Looping
Initial Start Delay -15,0
Initial Start Delay Variance 0,0
spawn_ennemy
Custom Event
spawn_bomb
Custom Event
Spawn_spécial
Custom Event
spawn_GOLD
Custom Event
1
BLUEPRINT
Ennemy In Game

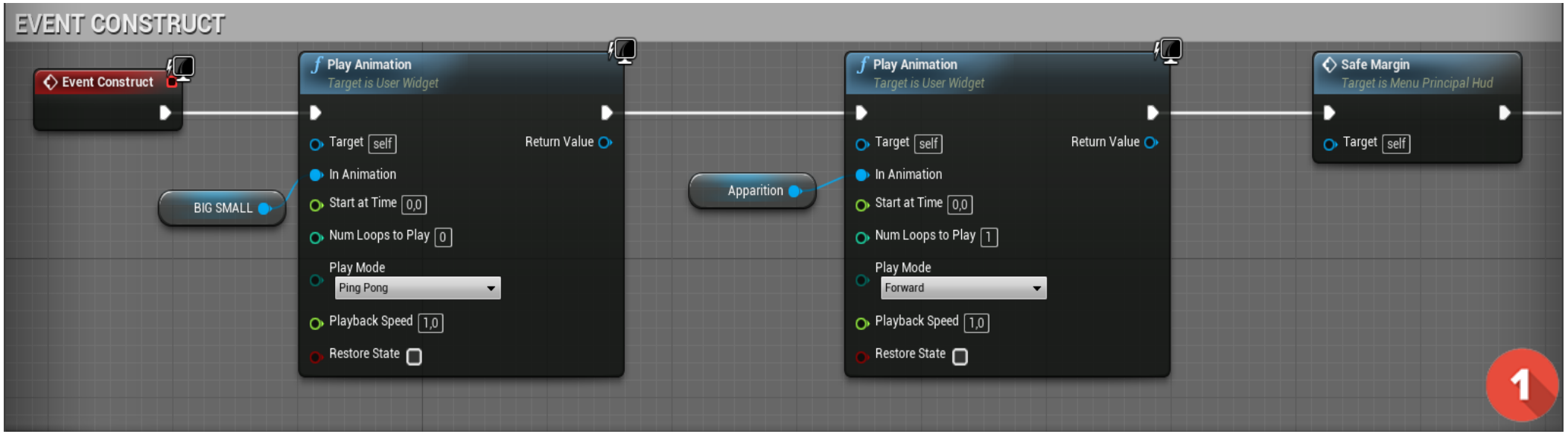
ANNEXE 14 : Blueprint Cooldness Awarness.

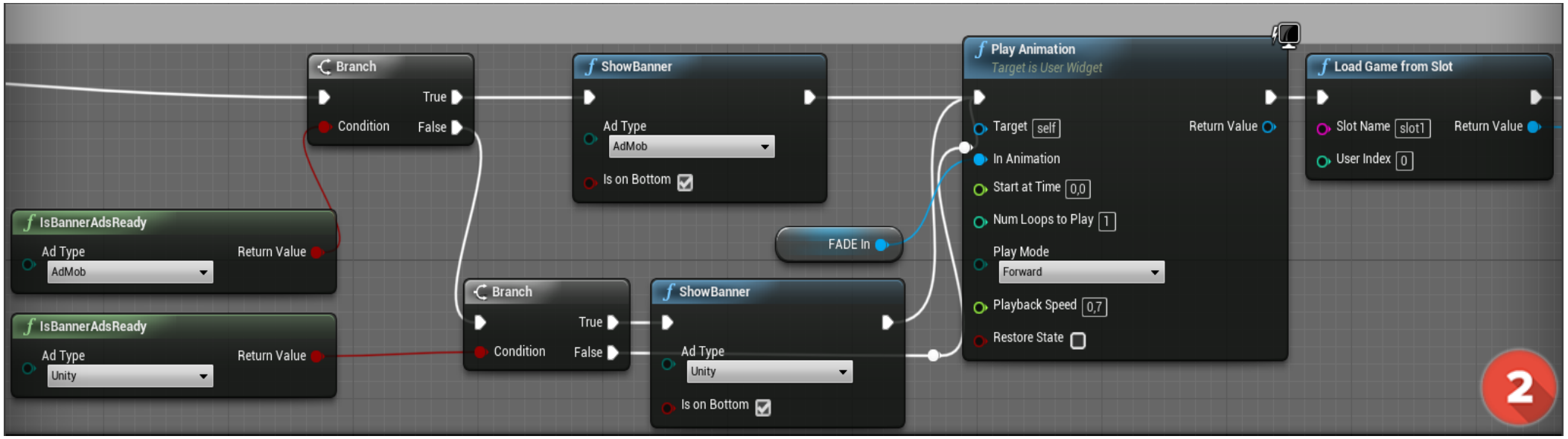

COOLDOWN AWARNES
Comment:
Cooldown Awareness
Branch
True
Condition
False
SET
Local Awareness Cooldown Rate
Awareness Cooldown Rate
Check Current AIState
In
Idle
Suspicious
Searching
Hostile
Awareness
0,0
AND
Add pin
State Locked
NOT
1

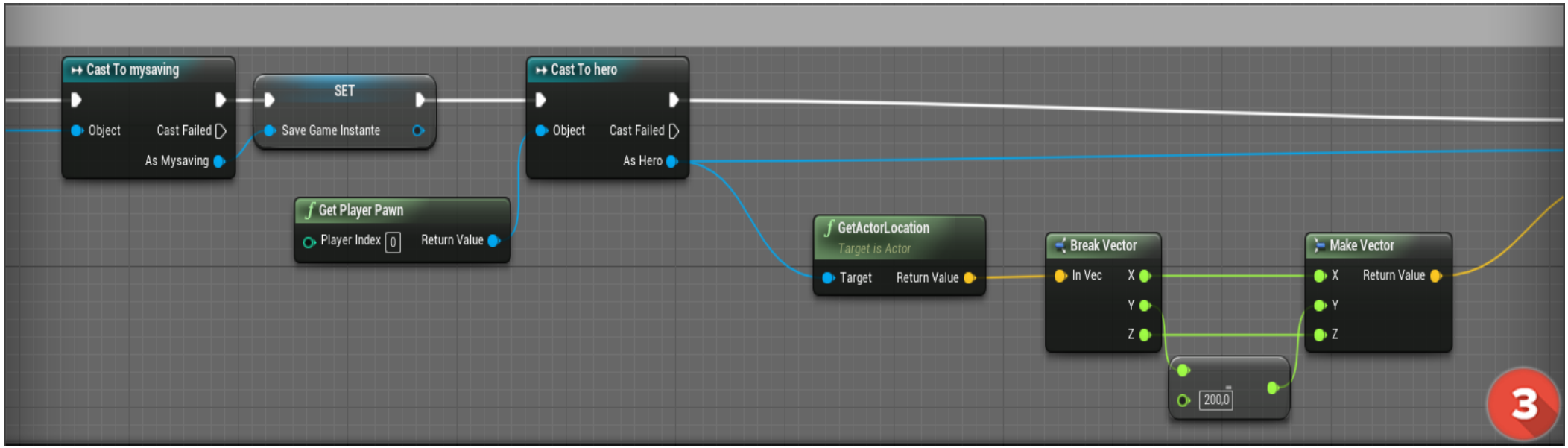

Branch
True
Condition
False
Enemy AIState Decreased
Awareness Cooldown Rate
0,25
Add pin
SET
Local Awareness Cooldown Rate
Awareness Cooldown Rate
SET
Local Awareness Cooldown Rate
Branch
True
Condition
False
Enemy AIState Decreased
Awareness Cooldown Rate
0,25
Add pin
SET
Local Awareness Cooldown Rate
Awareness Cooldown Rate
SET
Local Awareness Cooldown Rate
2

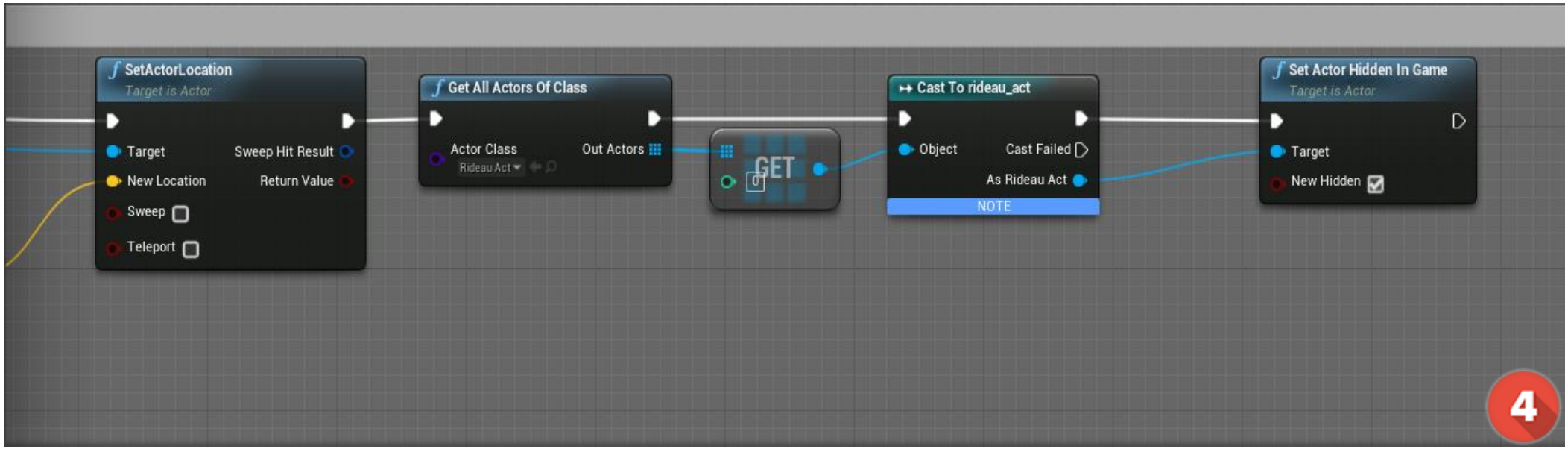

Awareness
Local Awareness Cooldown Rate
-
Max Awareness
Clamp (float)
Value
Return Value
Min
0,0
Max
SET
Awareness
Return Node
3

ANNEXE 15 : Fonction Event construct du menu principal.

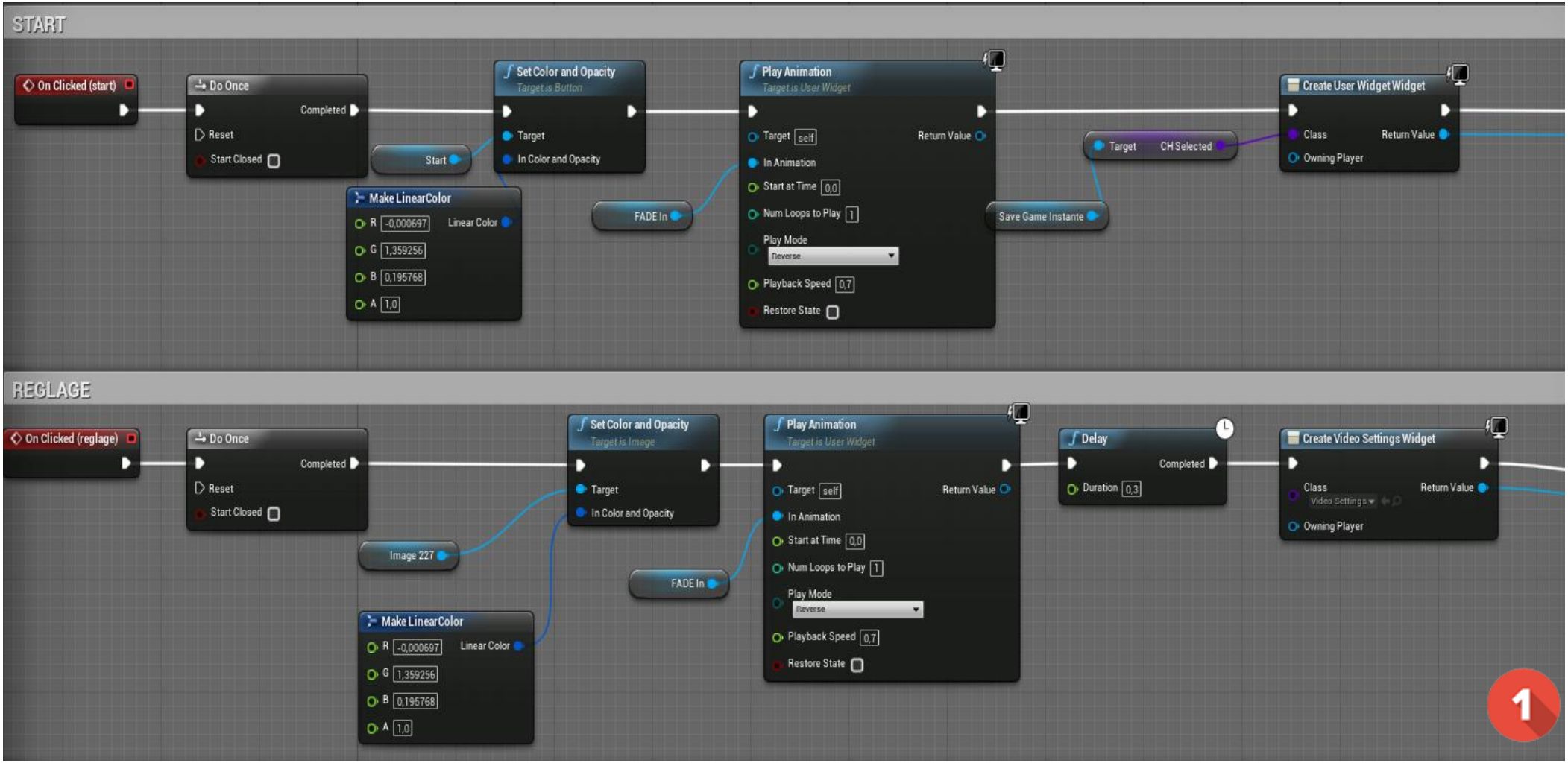

EVENT CONSTRUCT
Event Construct
Play Animation
Target is User Widget
Target self
Return Value
In Animation
BIG SMALL
Start at Time 0,0
Num Loops to Play 0
Play Mode
Ping Pong
Playback Speed 1,0
Restore State
Apparition
Play Animation
Target is User Widget
Target self
Return Value
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Forward
Playback Speed 1,0
Restore State
Safe Margin
Target is Menu Principal Hud
Target self
1

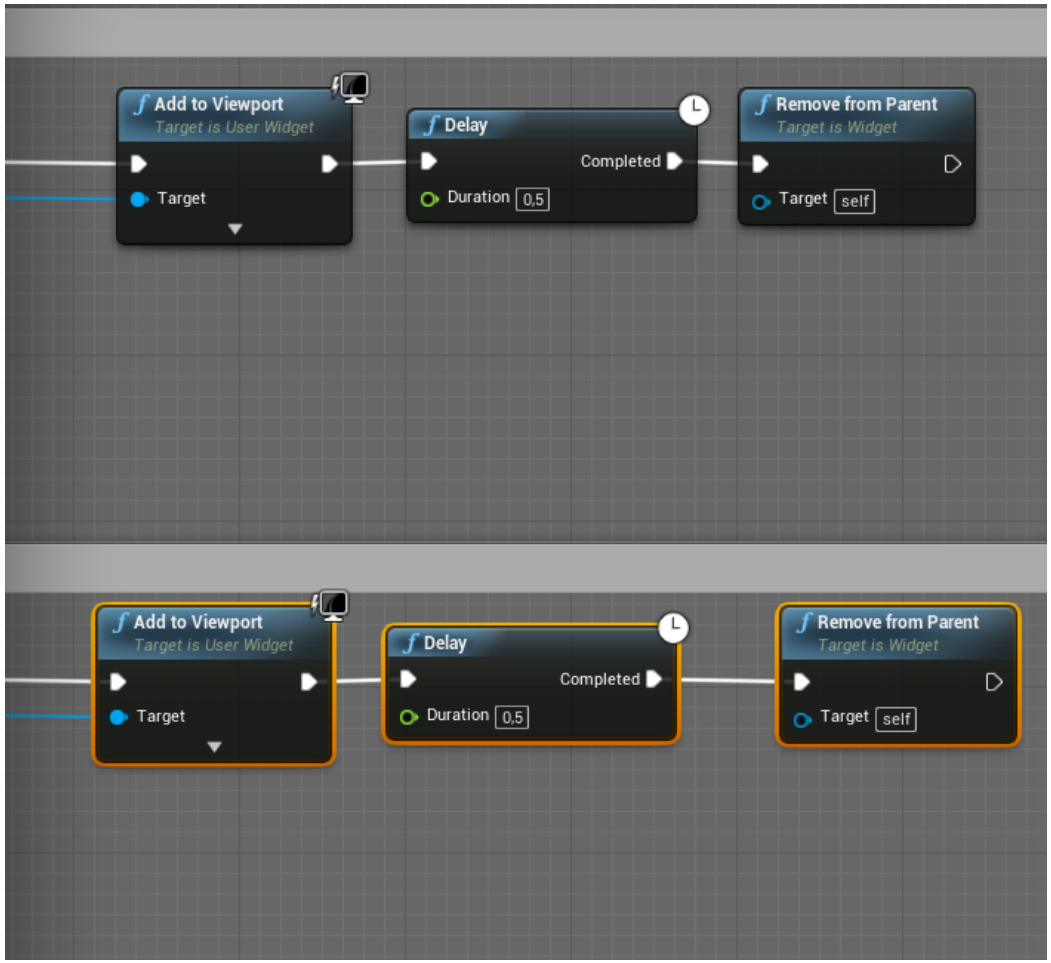

Branch
True
Condition
False
ShowBanner
Ad Type
AdMob
Is on Bottom
IsBannerAdsReady
Ad Type
AdMob
Return Value
IsBannerAdsReady
Ad Type
Unity
Return Value
Branch
True
Condition
False
ShowBanner
Ad Type
Unity
Is on Bottom
FADE In
Play Animation
Target is User Widget
Target self
Return Value
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Forward
Playback Speed 0,7
Restore State
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
2


Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Save Game Instante
Cast To hero
Object
Cast Failed
As Hero
Get Player Pawn
Player Index
0
Return Value
GetActorLocation
Target is Actor
Target
Return Value
Break Vector
In Vec
X
Y
Z
200,0
Make Vector
X
Y
Z
Return Value
3

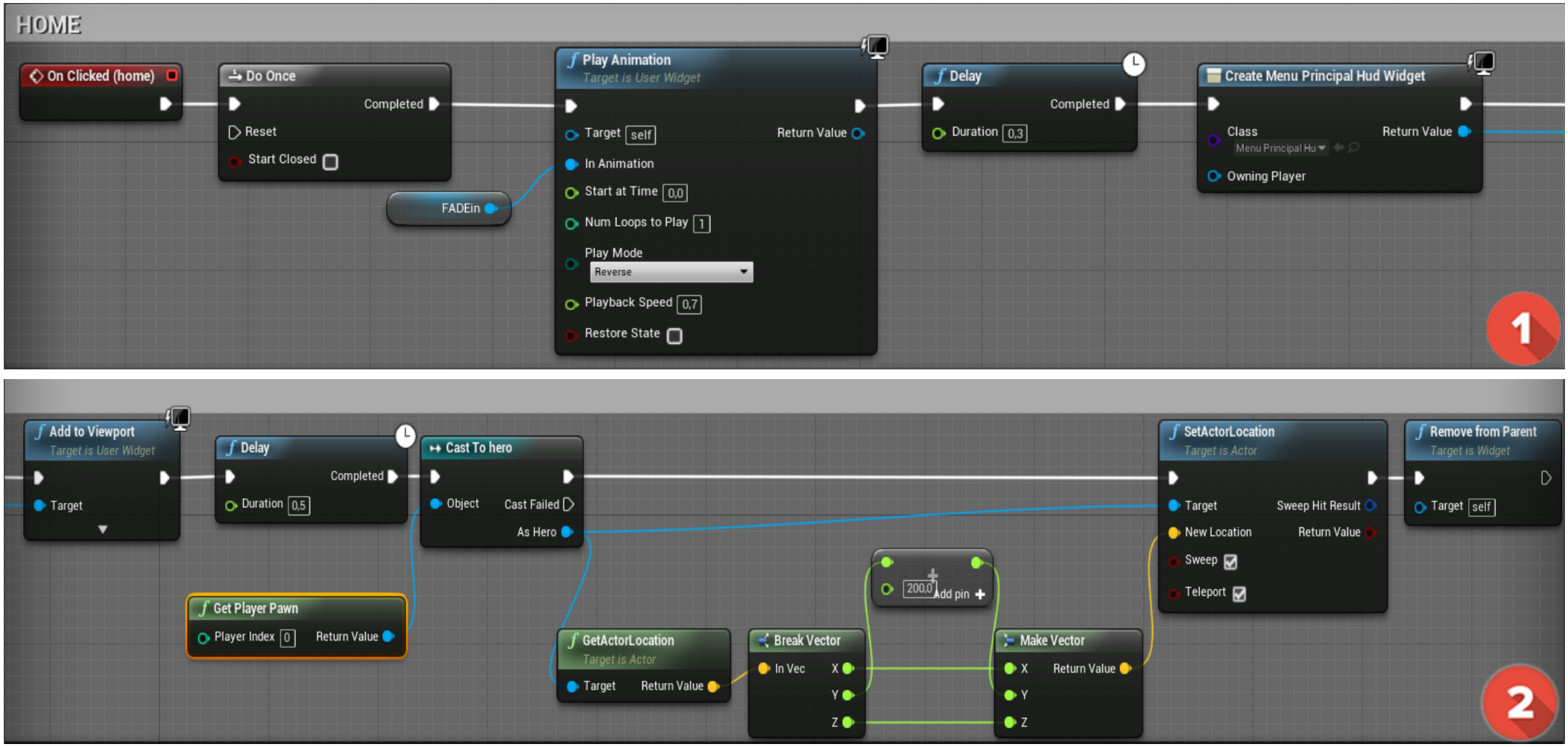

SetActorLocation
Target is Actor
Target
New Location
Sweep
Teleport
Sweep Hit Result
Return Value
Get All Actors Of Class
Actor Class
Rideau Act
Out Actors
GET
0
Cast To rideau_act
Object
Cast Failed
As Rideau Act
NOTE
Set Actor Hidden In Game
Target is Actor
Target
New Hidden
4

ANNEXE 16 : Blueprints Start et Réglage du menu principal.

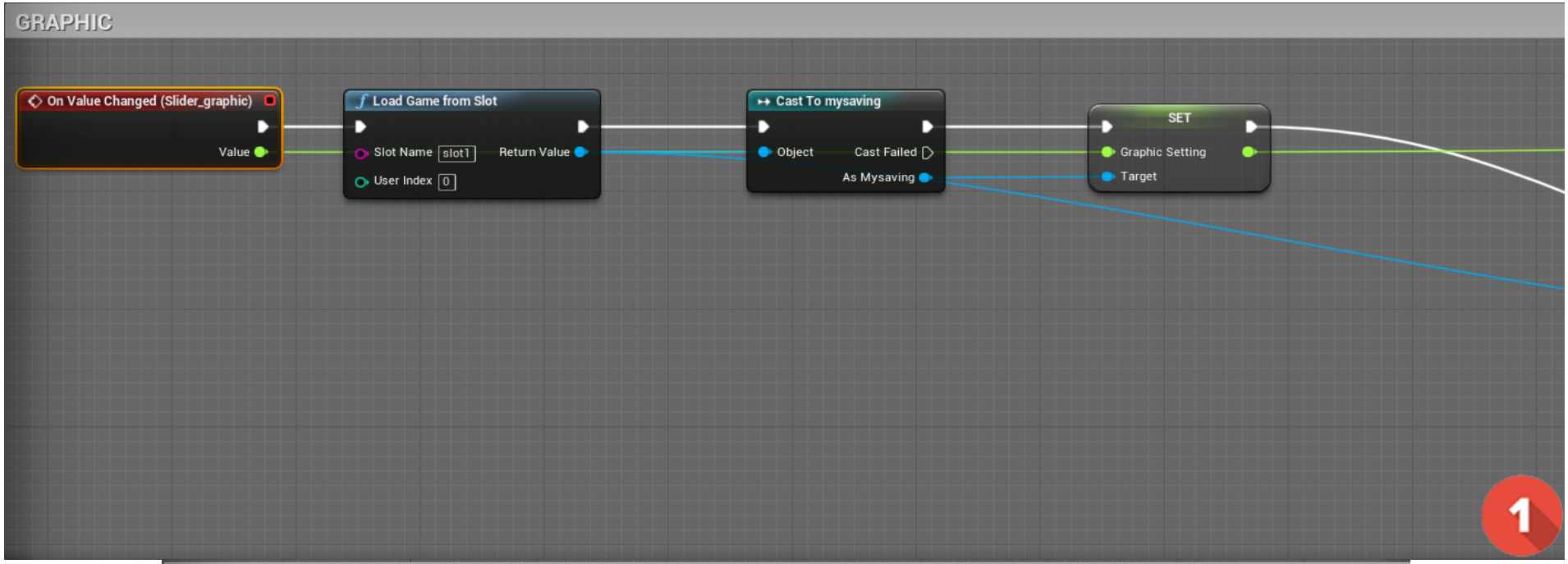

START
On Clicked (start)
Do Once
Completed
Reset
Start Closed
Start
Set Color and Opacity
Target is Button
Target
In Color and Opacity
Make LinearColor
R -0,000697
G 1,359256
B 0,195768
A 1,0
Linear Color
FADE In
Play Animation
Target is User Widget
Target self
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Reverse
Playback Speed 0,7
Restore State
Return Value
Save Game Instante
Target CH Selected
Create User Widget Widget
Class
Owning Player
Return Value
REGLAGE
On Clicked (reglage)
Do Once
Completed
Reset
Start Closed
Image 227
Set Color and Opacity
Target is Image
Target
In Color and Opacity
Make LinearColor
R -0,000697
G 1,359256
B 0,195768
A 1,0
Linear Color
FADE In
Play Animation
Target is User Widget
Target self
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Reverse
Playback Speed 0,7
Restore State
Return Value
Delay
Duration 0,3
Completed
Create Video Settings Widget
Class
Video Settings
Owning Player
Return Value
1

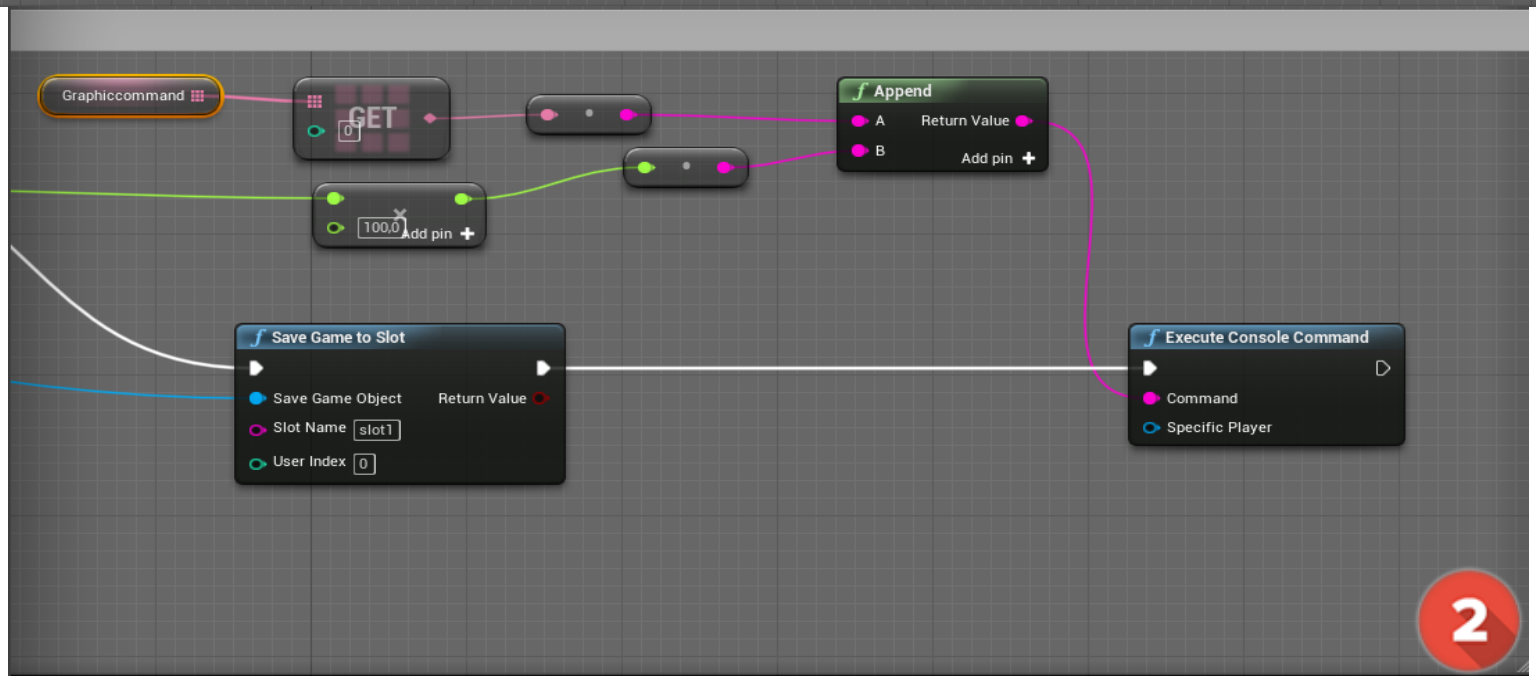

Add to Viewport
Target is User Widget
Target
Delay
Completed
Duration 0,5
Remove from Parent
Target is Widget
Target self
Add to Viewport
Target is User Widget
Target
Delay
Completed
Duration 0,5
Remove from Parent
Target is Widget
Target self

ANNEXE 17 : Blueprint Home du menu Réglage.

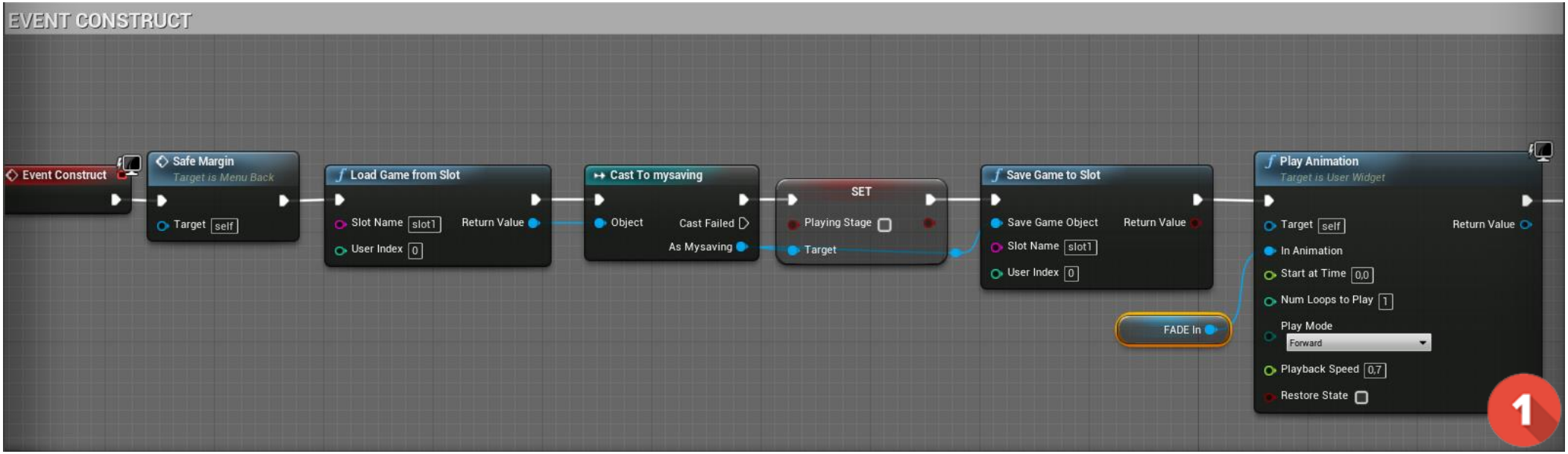

HOME
On Clicked (home)
Do Once
Completed
Reset
Start Closed
FADEin
Play Animation
Target is User Widget
Target self
Return Value
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Reverse
Playback Speed 0,7
Restore State
Delay
Completed
Duration 0,3
Create Menu Principal Hud Widget
Class
Menu Principal Hu
Return Value
Owning Player
1
Add to Viewport
Target is User Widget
Target
Delay
Completed
Duration 0,5
Cast To hero
Object
Cast Failed
As Hero
Get Player Pawn
Player Index 0
Return Value
GetActorLocation
Target is Actor
Target
Return Value
Break Vector
In Vec
X
Y
Z
200,0
Add pin
Make Vector
X
Y
Z
Return Value
SetActorLocation
Target is Actor
Target
Sweep Hit Result
New Location
Return Value
Sweep
Teleport
Remove from Parent
Target is Widget
Target self
2

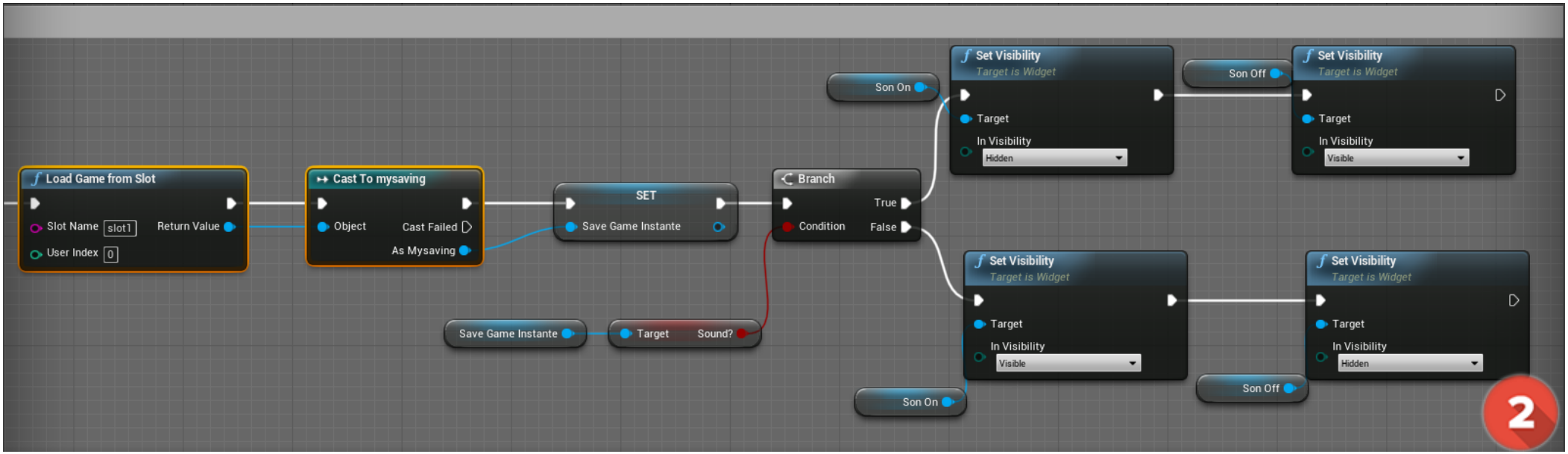
ANNEXE 18 : Fonction On value changed (Slider Graphic)


GRAPHIC
On Value Changed (Slider_graphic)
Value
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Graphic Setting
Target
1

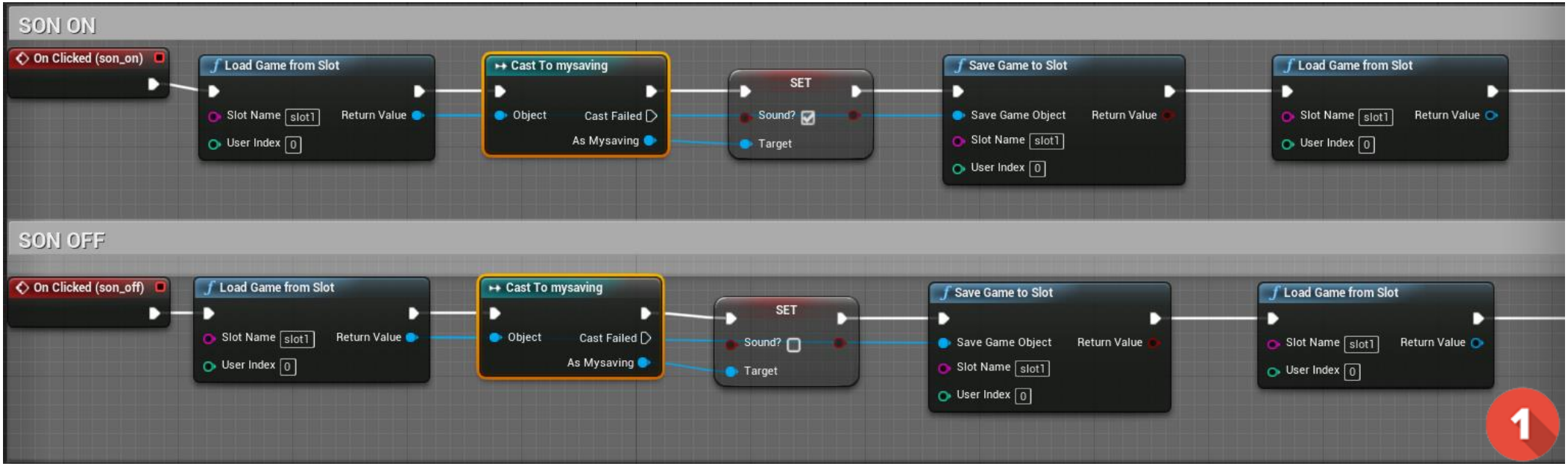

Graphiccommand
GET
0
Append
A
B
Return Value
Add pin
100.0
Add pin
Save Game to Slot
Save Game Object
Slot Name slot1
User Index 0
Return Value
Execute Console Command
Command
Specific Player
2

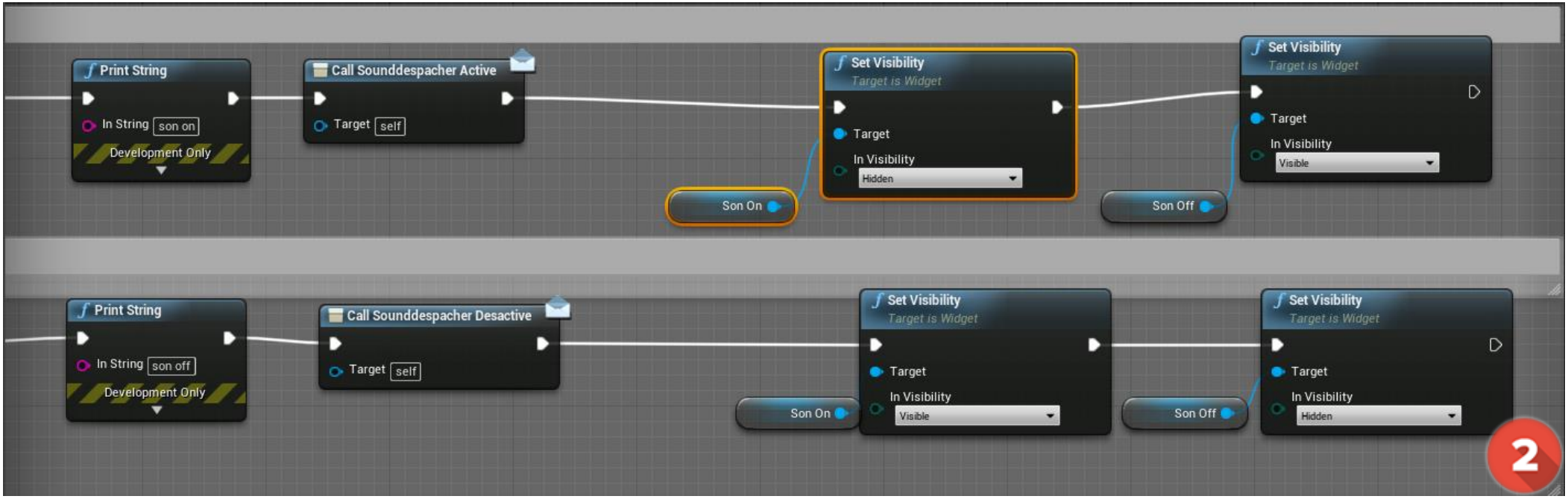
ANNEXE 19 : Blueprint Event Construct du menu back.


EVENT CONSTRUCT
Event Construct
Safe Margin
Target is Menu Back
Target self
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Playing Stage
Target
Save Game to Slot
Save Game Object
Return Value
Slot Name slot1
User Index 0
FADE In
Play Animation
Target is User Widget
Target self
Return Value
In Animation
Start at Time 0,0
Num Loops to Play 1
Play Mode
Forward
Playback Speed 0,7
Restore State
1

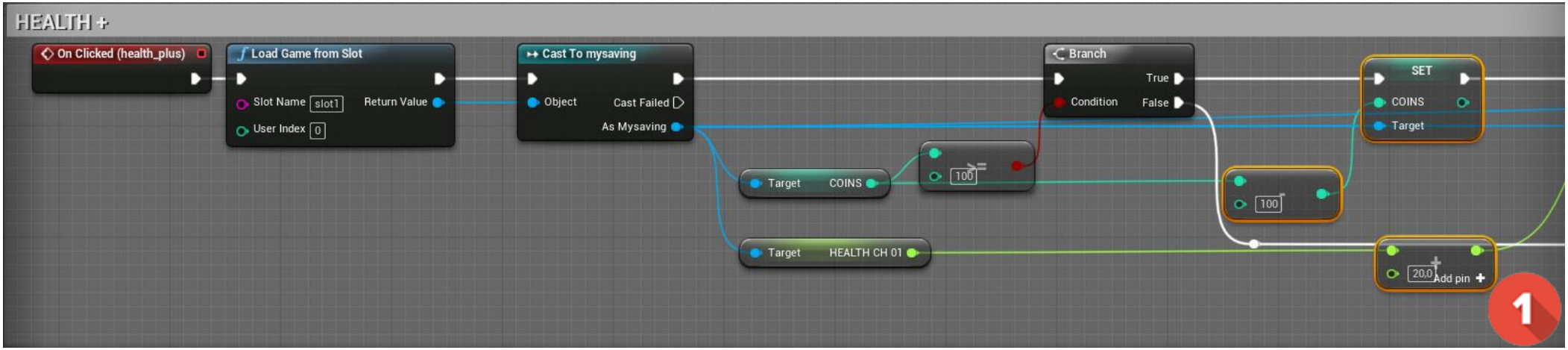

Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Save Game Instante
Branch
True
Condition
False
Son On
Set Visibility
Target is Widget
Target
In Visibility
Hidden
Son Off
Set Visibility
Target is Widget
Target
In Visibility
Visible
Save Game Instante
Target
Sound?
Set Visibility
Target is Widget
Target
In Visibility
Visible
Son On
Set Visibility
Target is Widget
Target
In Visibility
Hidden
Son Off
2

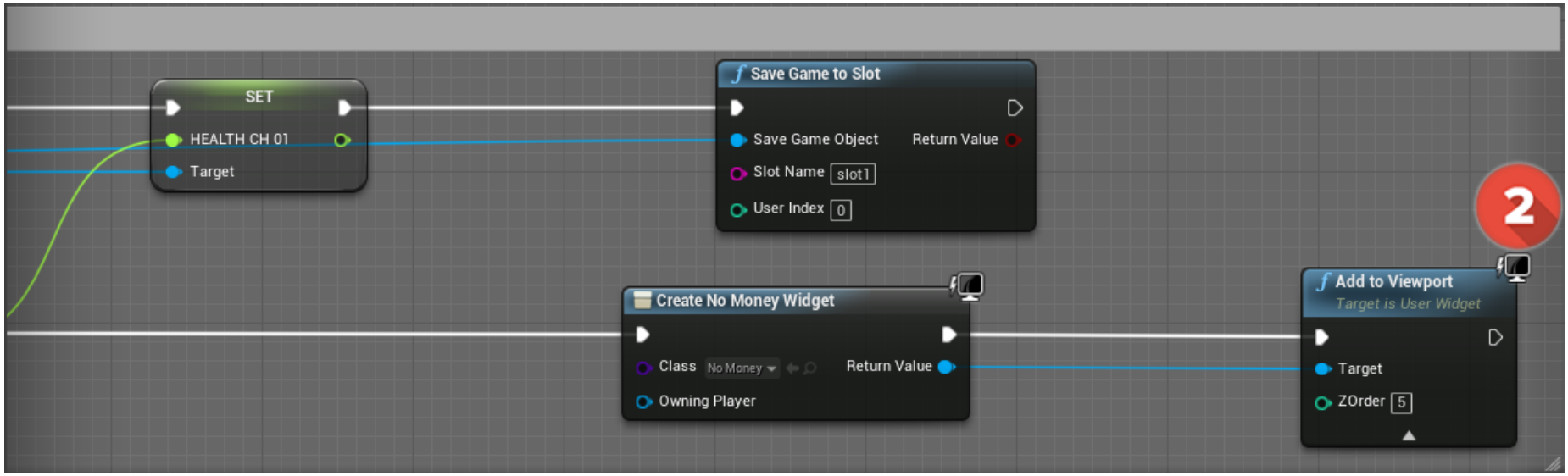
ANNEXE 20 : Blueprints On Clicked (son_on) et On Clicked (son_off)

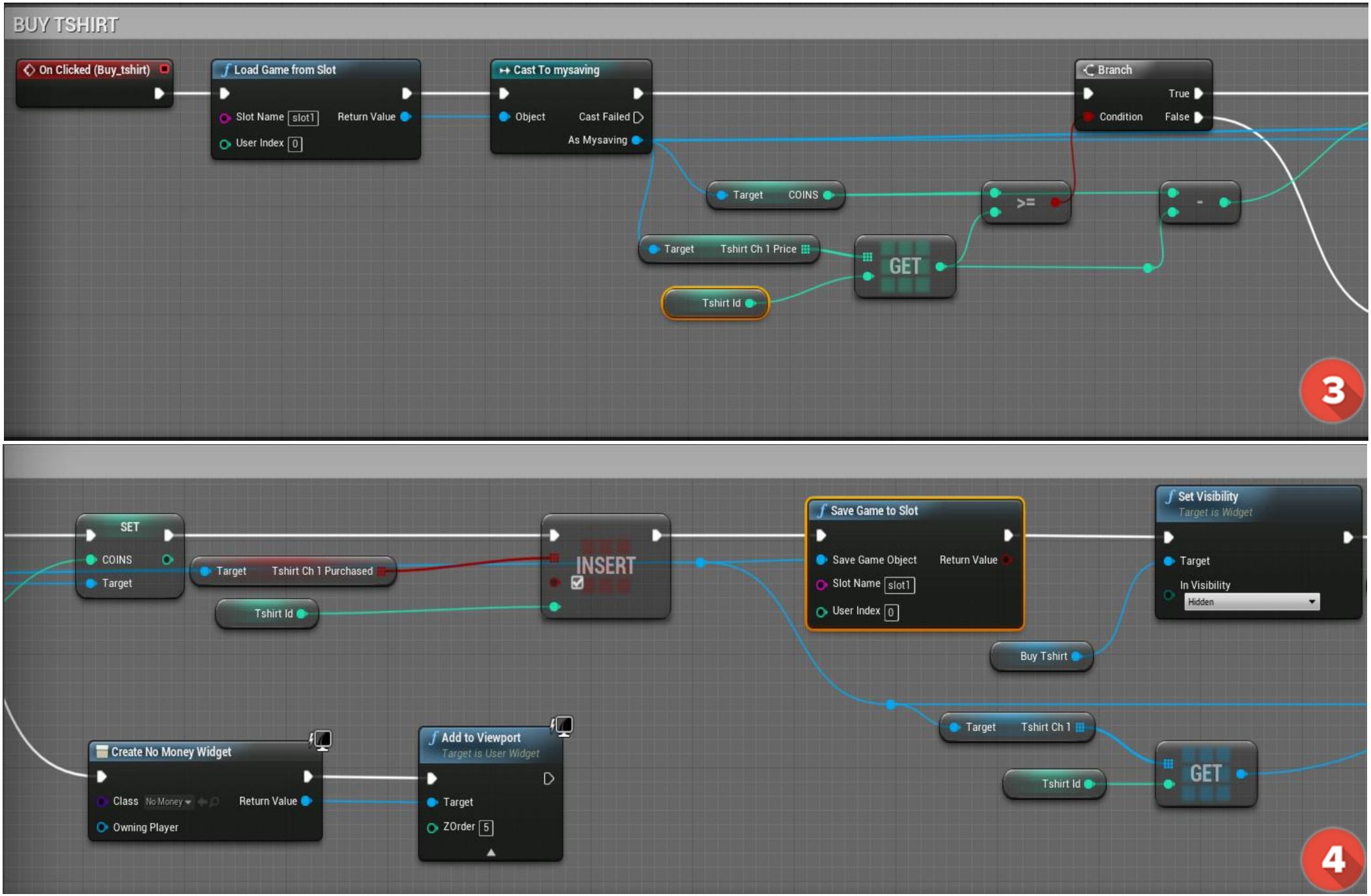

SON ON
On Clicked (son_on)
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Sound?
Target
Save Game to Slot
Save Game Object
Slot Name slot1
User Index 0
Return Value
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
SON OFF
On Clicked (son_off)
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
Cast To mysaving
Object
Cast Failed
As Mysaving
SET
Sound?
Target
Save Game to Slot
Save Game Object
Slot Name slot1
User Index 0
Return Value
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
1

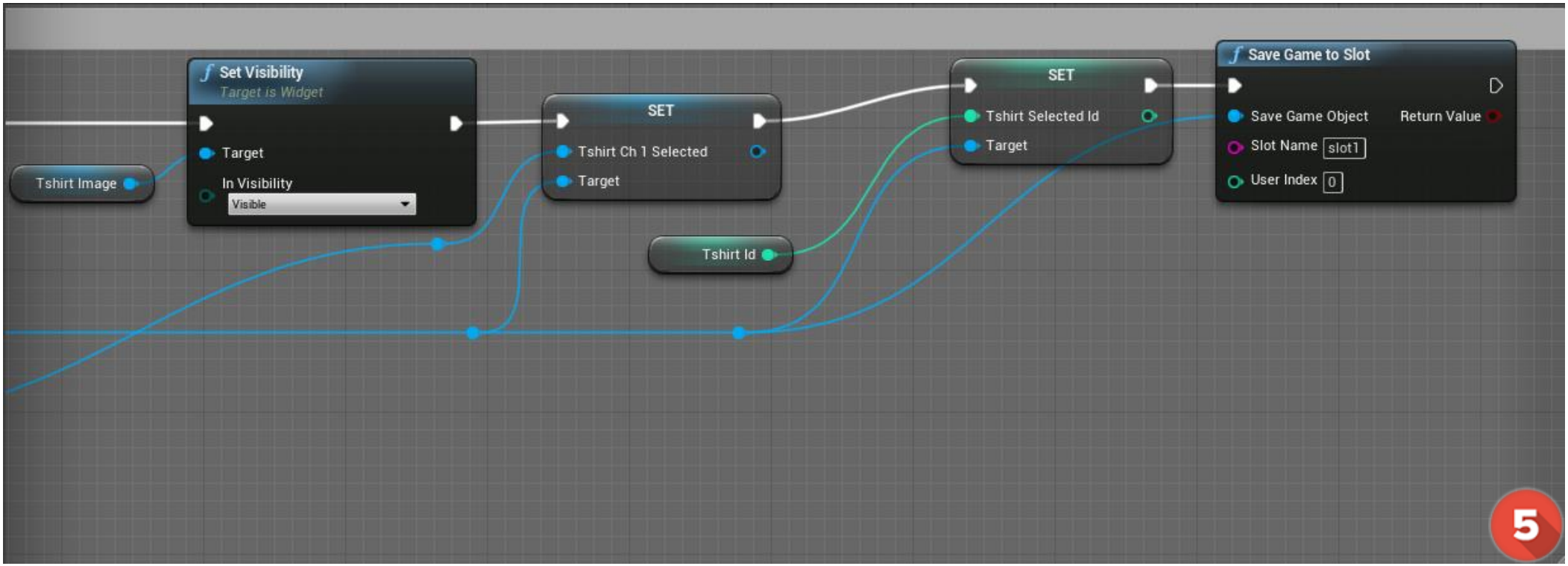

Print String
In String son on
Development Only
Call Sounddespacher Active
Target self
Son On
Set Visibility
Target is Widget
Target
In Visibility
Hidden
Son Off
Set Visibility
Target is Widget
Target
In Visibility
Visible
Print String
In String son off
Development Only
Call Sounddespacher Desactive
Target self
Son On
Set Visibility
Target is Widget
Target
In Visibility
Visible
Son Off
Set Visibility
Target is Widget
Target
In Visibility
Hidden
2

ANNEXE 21 : Blueprint Health+ et Buy Tshirt


HEALTH +
On Clicked (health_plus)
Load Game from Slot
Slot Name slot1
User Index 0
Return Value
Cast To mysaving
Object
Cast Failed
As Mysaving
Target COINS
100
Branch
Condition
True
False
100
SET
COINS
Target
Target HEALTH CH 01
20,0
Add pin
1


SET
HEALTH CH 01
Target
Save Game to Slot
Save Game Object
Return Value
Slot Name slot1
User Index 0
Create No Money Widget
Class No Money
Owning Player
Return Value
Add to Viewport
Target is User Widget
Target
ZOrder 5
2


BUY TSHIRT
On Clicked (Buy_tshirt)
Load Game from Slot
Slot Name slot1
Return Value
User Index 0
Cast To mysaving
Object
Cast Failed
As Mysaving
Branch
True
Condition
False
Target COINS
>=
-
Target Tshirt Ch 1 Price
GET
Tshirt Id
3
SET
COINS
Target
Target Tshirt Ch 1 Purchased
Tshirt Id
INSERT
Save Game to Slot
Save Game Object
Return Value
Slot Name slot1
User Index 0
Set Visibility
Target is Widget
Target
In Visibility
Hidden
Buy Tshirt
Target Tshirt Ch 1
Tshirt Id
GET
Create No Money Widget
Class No Money
Return Value
Owning Player
Add to Viewport
Target is User Widget
Target
ZOrder 5
4


Set Visibility
Target is Widget
Target
In Visibility
Visible
Tshirt Image
SET
Tshirt Ch 1 Selected
Target
Tshirt Id
SET
Tshirt Selected Id
Target
Save Game to Slot
Save Game Object
Return Value
Slot Name
slot1
User Index
0
5