

RAPPORT DE PROJET

 Février 2024 – Mai 2024

**COSTA Julien / JOURNOUD Lucas**

Université Lyon 2 – L3 Informatique

Enseignant référant : Albert Merino Gomez

# Introduction

Table des matières

[Introduction 2](#_Toc166813718)

[Synopsis 4](#_Toc166813719)

[Méthodologie 5](#_Toc166813720)

[Répartition des tâches 5](#_Toc166813721)

[JOURNOUD 5](#_Toc166813722)

[COSTA 5](#_Toc166813723)

[Logiciels utilisés 5](#_Toc166813724)

[Choix techniques 5](#_Toc166813725)

[Développement 6](#_Toc166813726)

[JOURNOUD Lucas 6](#_Toc166813727)

[Création du Projet 6](#_Toc166813728)

[Convention 7](#_Toc166813729)

[Commande de jeu 7](#_Toc166813730)

[Création du personnage 8](#_Toc166813731)

[Création des blocks 9](#_Toc166813732)

[COSTA Julien 12](#_Toc166813733)

[Conclusions 19](#_Toc166813734)

[Annexes 20](#_Toc166813735)

[Annexe 1 20](#_Toc166813736)

[Annexe 2 21](#_Toc166813737)

# Synopsis

Dans Wario Bros, les joueurs incarnent l’anti-héros, Wario, dans une aventure à travers des mondes et des niveaux afin d’affronter son rival Mario. Contrairement à Mario, Wario n'hésite pas à utiliser la force brute et la ruse pour parvenir à ses fins. Au fur et à mesure qu'il progresse, Wario affronte différents monstres sur son chemin afin de s’améliorer, tout en cherchant à amasser autant de pièces que possible pour construire son propre empire.

# Méthodologie

## Répartition des tâches

Pour la répartition des tâches nous avons décidé de s’occuper chacun de ce que l’ont maitrisais le mieux, est ce qui était en accord avec notre projet professionnel. Dans ce contexte, nous avons procédé comme ceci :

### JOURNOUD

JOURNOUD Lucas s’occupe de la partie technique du projet. Cela comprend, la création du Git, du projet Unreal Engine et de toute la partie fonctionnelle des différents éléments construit dans Unreal Engine (gestion des mouvements, des actions, des fonctionnalités, hitbox[[1]](#footnote-1), Gestion de l’UI[[2]](#footnote-2) etc…).

### COSTA

COSTA Julien lui de son côté, s’occupe de la partie graphique du jeu. C’est-à-dire des représentations des différents éléments qui compose le jeu comme les Sprites[[3]](#footnote-3), les Animation 2D (Flipbook), les matériaux et la création du niveau.

## Logiciels utilisés

Pour la création du jeu nous avons utilisé [Unreal Engine 5.3](https://www.unrealengine.com/fr) un moteur de jeu célèbre et gratuit[[4]](#footnote-4).

Pour la gestion des fichiers nous avons utilisés [Github Desktop](https://desktop.github.com/) étant un logiciel de version de control en interface graphique basé sur [Git](https://git-scm.com/) et développé par [Github](https://github.com/).

Pour la création de Sprites nous utilisons [Aseprite](https://store.steampowered.com/app/431730/Aseprite/) un outil payant de pixel-art pour créer des animations 2D, des Sprites, et tout type de graphique pour les jeux.

## Choix techniques

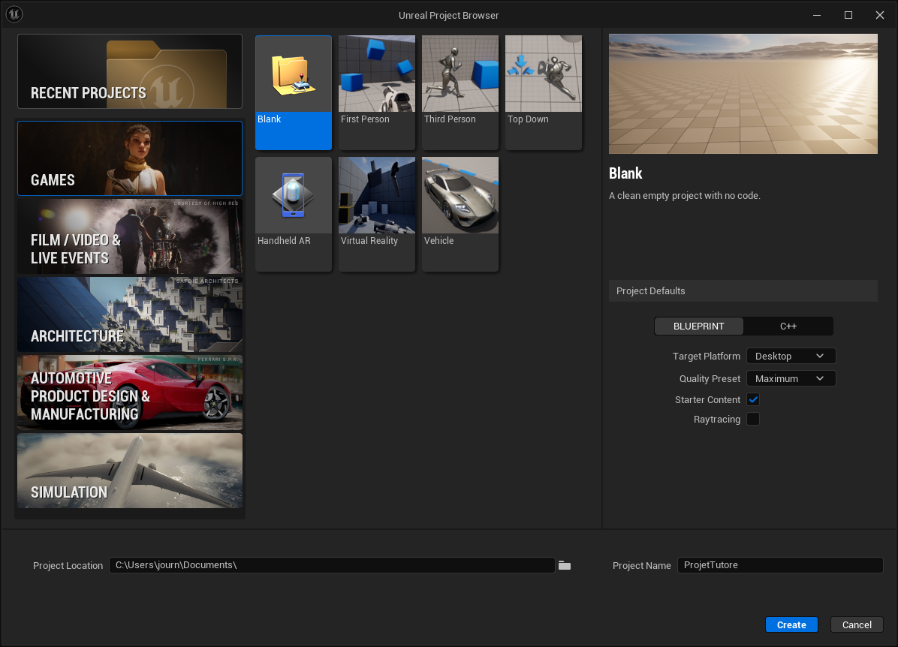
Lors de la réalisation de ce projet nous avons eu à faire plusieurs choix techniques. Pour commencer, nous avons décidé d’utiliser Unreal Engine 5.3 étant la toute dernière version disponible d’Unreal Engine a la création du projet. Cela afin de commencer notre apprentissage sur des outils à jour et non obsolète. Ensuite nous avons dû faire un choix quand au jeu que nous allions créer. Il était déjà prévu de faire un jeu Mario Bros, mais évidemment nous n’allions pas re créer le jeu en entier. Nous avons donc décidé de recréer certains éléments iconiques de la franchise Mario Bros et de créer 1 seul niveau de jeu. Nous aurions pu en créer d’autre sans problèmes. En effet, une fois les différents élément composant un niveau créer, il est très facile de créer un niveau. Il nous suffit pour cela juste de placer les différents éléments dans le monde. Nous aurions donc aisément pu en créer d’autre. Cela aurait juste impliqué la création d’un menu pour choisir le niveau à lancer. Notre but n’étant pas de re créer le jeu en entier, et ayant des contrainte de temps nous avons donc décider d’en créer qu’un seul.

# Développement

## JOURNOUD Lucas

S’occupant de la partie fonctionnelle du jeu j’ai eu à faire plusieurs choix techniques et à résoudre plusieurs problèmes auquel j’ai pu être confronté. Je vais vous en présenter certains ci-dessous.

### Création du Projet

Pour commencer j’ai dû créer le projet Unreal Engine. J’ai donc choisi la catégorie jeu. Ensuite différente Template existe dépendamment du type de jeu voulue. Aucun n’était présent pour les jeux 2D, je suis donc partie sur un projet vide. Ensuite il y avait quelques options à choisir pour la création du jeu. La première d’entre elle était le choix entre C++ ou Blueprint. En effet Unreal Engine propose 2 méthodes de développement de jeu vidéo. Les Blueprints et le C++. Le Blueprint étant un langage de script visuel ce qui le rend plus accessible pour les personnes n’étant pas familières avec la programmation traditionnelle. Le C++ offre lui des performances plus élevées et n’as pas de limite fonctionnelle, en effet certains fonctions avancés et optimisations spécifique ne peuvent être réalisé de manière efficace avec Blueprints seul ([voir Annexe 1](#_Annexe_1)).

Interface de création de projet Unreal Engine

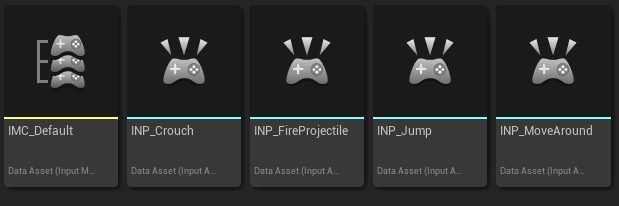
Les 2 peuvent être utiliser pour la création du même jeu indépendamment du choix fait à la création, mais Unreal Engine créera les différentes structures de fichier approprié par rapport a ce choix initial. Je voulais initialement choisir le C++, étant mon langage préféré et le maitrisant plutôt bien depuis plusieurs années j’ai d’abord pensé à faire ce choix. Malheureusement le C++ étant plus complexe que les Blueprints, tous les tutoriels pour débutant sur Unreal Engine que j’ai pu trouver n’aborde que la création de jeu à travers les Blueprints et non grâce au C++. Je me suis donc résigné à utiliser les Blueprints ce qui me permettra en même temps d’apprendre à les utiliser et de ne pas rester sur mes acquis en utilisant le C++. Je devais ensuite choisir la plateforme cible du jeu. Nous développons un jeu PC, j’ai donc choisi Desktop. Pour la qualité graphique j’ai choisi Maximum. Etant donné que nous développons un jeu ayant des graphismes Low poly cela n’impactera pas les performances. Je n’ai pas activé le Raytracing[[5]](#footnote-5) pour les mêmes raisons.

### Convention

Une fois le projet créer, il fallait que choisisse comment organiser le projet, nommer mes variables etc… Pour cela, j’ai décidé d’utiliser le [Allar’s Style Guide (UE5 v2)](https://github.com/Allar/ue5-style-guide/blob/v2/README.md) étant une des conventions de nommage de fichier et d’organisation de ceci les plus rependu pour Unreal Engine 4 et 5. Celui-ci implique plusieurs choses notamment l’utilisation du [PascalCase](https://github.com/Allar/ue5-style-guide/blob/v2/README.md#211-always-use-pascalcase) pour le nommage des variables, [l’utilisation de préfixe / suffixe](https://github.com/Allar/ue5-style-guide/blob/v2/README.md#1-asset-naming-conventions) dans le nommage des fichiers (par exemple pour les Blueprint « BP\_ ») et l’organisation des fichiers selon une [structure bien définie](https://github.com/Allar/ue5-style-guide/blob/v2/README.md#2e1-example-project-content-structure). J’ai aussi pris comme décision que tout les commentaires, nom de variables etc… seraient en anglais car ceci est la langue utiliser de partout dans le développement de jeu vidéo.

### Commande de jeu

Une fois arrivé sur l’éditeur Unreal Engine la première chose que j’ai voulu faire était la création du personnage. Mais pour cela il fallait d’abord créer les différentes commandes du personnage. J’ai donc fait une liste des commandes que le joueur allait pouvoir utiliser. Celle la comprenait le fait de pouvoir sauter, bouger, se baisser et lancer des boules de feu. Pour faire cela j’utilise le « Enhanced Input system » étant le nouveau système de gestion de commandes de Unreal Engine 5.

Pour cela il me faut juste créer mes différentes actions et les « mapper » à mes différents boutons dans mon fichier « IMC\_Default ». Ensuite dés que l’on appuiera sur un de ces boutons, cela déclenchera l’action approprié ([voir Annexe 2](#_Annexe_2)).

Différentes actions du jeu Wario Bros

Voulant que le jeu soit le plus accessible possible j’ai décidé d’ajouté le support manette au jeu. Pour cela rien de plus simple, il me suffit d’aller dans mon fichier « IMC\_Default » et sous l’action « Jump » par exemple de rajouter le bouton de ma manette. Il faut savoir qu’Unreal Engine 5.3 ne fournit qu’un support pour les manettes Xbox. En effet cela est dû à l’API[[6]](#footnote-6) Windows qui ne prend pas en charges les autres types de contrôleur (sans logiciel externe). Cela a été changer très récemment par Windows qui prend finalement en charge les autres types de contrôleur comme les manette Playstation mais cela n’a été ajouté que dans Unreal Engine 5.4 sortie après le début de notre projet. Il nous suffirait donc de passer à la nouvelle version d’Unreal Engine pour que bon nombre de contrôleurs sois aussi pris en charge. Nous ne l’avons pas fait car la version stable d’Unreal Engine 5.4 est sortie vers la fin de notre projet et nous n’avons pas d’autre type de manette afin de pouvoir effectuer des tests et vérifier que cela fonctionne. Je ne le savais pas à ce moment-là mais le support manette que j’avais ajouté ne serais que partiel. En effet plus tard lors de la création du menu pour relancer le jeu ou quitter, ces boutons doivent être cliquer à la souris et il n’y a pas de moyen « simple » pour que cela fonctionne à la manette. Le support manette fonctionne donc en jeu mais non dans le menu.

### Création du personnage

Une fois les commandes créer j’ai pu passer à la création du personnage. Pour cela j’ai créé énormément de fonction. Je ne peux pas tout expliquer ici, alors je vais vous en détailler une seule (c’est un peu un anachronisme car c’est une fonction du personnage que j’ai créé bien plus tard, utilisant des éléments n’existant pas au début de la création du jeu).

Fonction « Throw Fireball »

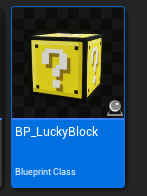
Cette fonction « Throw Fireball » permet au joueur de lancer des boules de feu. Pour commencer, on vérifie que l’évolution du joueur est égale à 2. En effet j’ai mis en place une variable évolution. Wario possédant différente forme : « petit » (valeur 0), « grand » (valeur 1) et « Wario de feu » (valeur 2) il fallait une variable pour stocker ceci. Je vérifie donc que Wario sois en forme Wario de feu afin de savoir s’il est capable de lancer des boules de feu. Ensuite le reste du code est dans une fonction « Do Once » celui-ci permet de faire en sorte que ce code ne soit exécuté qu’une fois. Sans ceci le joueur pourrait lancer autant de boule de feu qu’il souhaite à la seconde. A l’intérieur du « Do Once » je fais apparaître ma boule de feu que j’ai codé précédemment au niveau de Wario. Je n’ai pas besoin de gérer ses déplacements ni son comportement ici car ceci est gérer dans le Blueprint de ma boule de feu. Ici la seule chose que je fais c’est de faire apparaître une boule de feu, la boule de feu est autonome et suis ensuite son propre comportement. Ensuite je lance le son de la boule de feu. Nous avons pu le [récupérer sur internet](https://themushroomkingdom.net/sounds.shtml), les différents sons des anciens jeux Mario bros étant récupérable facilement. Une fois ceci fait je ne peux pas arrêter le code la dessus car dans l’état actuel le joueur pourrais lancer une boule de feu et ne plus jamais en lancer ! Après avoir lancer le son j’appelle donc une fonction que je viens de créer s’appelant « TriggerColdown ». Cette fonction va attendre 0.5s et réactiver le « Do Once » permettant au joueur de pouvoir relancer des boules de feu. Voici un récapitulatif de l’algorithme.

Vérifier que Wario peux lancer des boules de feu -> faire apparaitre une boule de feu -> jouer le son de la boule de feu -> attendre 0.5s -> permettre au joueur de pouvoir relancer une boule de feu

### Création des blocks

Pour la création de notre jeu il fallait créer différents blocks avec lequel le joueur devait pouvoir interagir. Je vais vous présenter une partie de son fonctionnement.

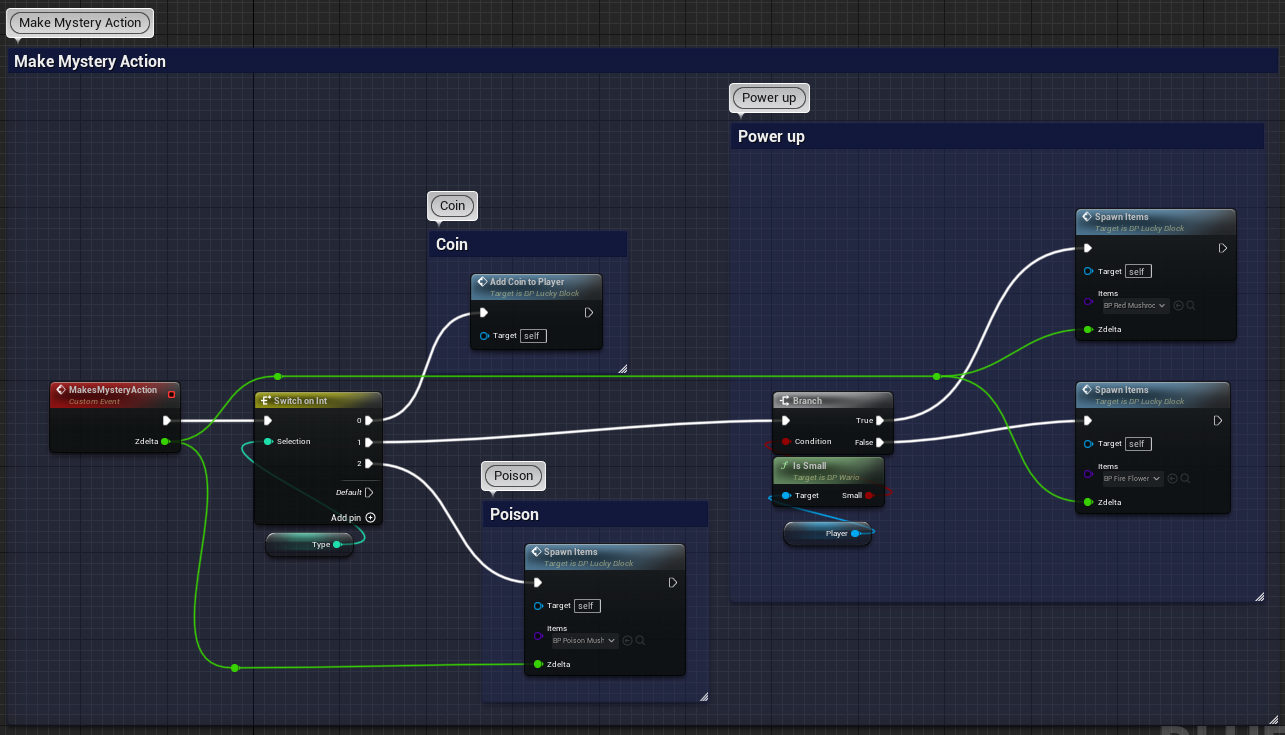
#### Contenu du block

Si vous avez déjà jouer à un jeu Mario Bros vous devriez savoir que Wario peut les taper, afin que ceci dévoile leur contenu mystère. Notamment des pièces ou des power-up comme des champignons ou des fleurs de feu. Afin de faire en sorte que mon collègue COSTA Julien puisse choisir facilement le contenu du block quand il les place dans le niveau j’ai mis en place le système suivant sur les blocks. Il peut choisir leur type directement depuis l’éditeur, il lui suffit de cliquer sur le block qu’il a rajouter dans le niveau et il peut choisir son type grâce à cette variable :



Il lui suffit de changer ce nombre pour changer le contenu !

Pour cela c’est tout simple, j’ai créé une fonction « MakeMysteryAction » appelé dès lors que le block est tapé par Wario.



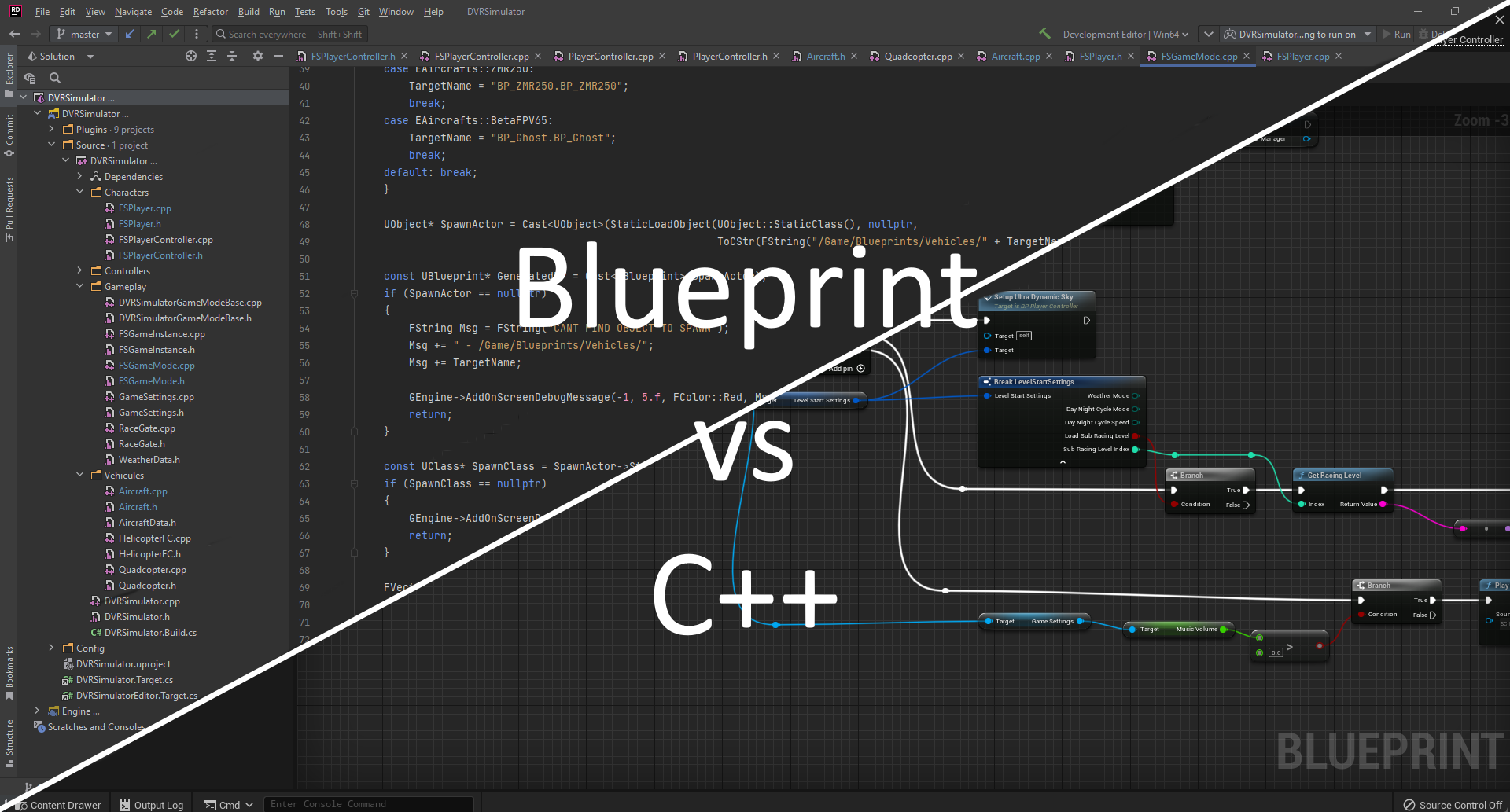
Quand cette fonction est appelée cela vérifie la valeur de la variable Type dont nous parlions avant. Si celle-ci est égale a 0, alors on appelle la fonction permettant d’ajouter 1 pièce d’or a Wario. Si elle est égale a 2 on vérifie l’évolution de Wario. Si c’est petit Wario on appelle la fonction permettant de faire apparaître un champignon et s’il est déjà grand la fonction permettant de faire apparaître la fleur de feu. Dans le cas ou le Type est égale a 2 alors on appelle la fonction permettant de faire apparaître le champignon vénéneux infligeant des dégâts a Wario s’il le récupère.

## COSTA Julien

# Conclusions

# Annexes

## Annexe 1



Blueprint vs C++ (en haut à gauche le C++ et en bas a droit les blueprints)

## Annexe 2

Input Mapping Context du jeu

1. Le masque de collision (ou hitbox) est un terme spécifique aux jeux vidéo qui définit la zone sensible d'un élément de jeu (les contrôles de collision vous permettent de déterminer quand les objets entrent en contact ou se chevauchent). [↑](#footnote-ref-1)
2. UI est l'acronyme anglais de ”User Interface”. L'UI Design désigne la conception de l'interface utilisateur. [↑](#footnote-ref-2)
3. Un Sprite est un maillage planaire texturé et un matériau associé qui peuvent être rendus dans le monde. [↑](#footnote-ref-3)
4. Unreal Engine est gratuit pour les développeurs de jeux, à condition que leur produit rapporte moins d'un million de dollars de recettes brutes. Au-delà, les développeurs doivent payer une redevance de 0,05 $ par vente. [↑](#footnote-ref-4)
5. Le ray tracing est une technologie qui simule le comportement physique des rayons lumineux, leurs réflexions et leurs interactions avec l'environnement. Cela permet d'améliorer considérablement l'émulation de l'éclairage et d'obtenir des rendus 3D plus réalistes et plus agréables à voir [↑](#footnote-ref-5)
6. Une API (application programming interface ou « interface de programmation d'application ») est une interface logicielle qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités. [↑](#footnote-ref-6)