

RAPPORT DE PROJET

 Février 2024 – Mai 2024

**COSTA Julien / JOURNOUD Lucas**

Université Lyon 2 – L3 Informatique

Enseignant référant : Albert Merino Gomez

# Introduction

Dans le cadre de notre projet de fin d'études, nous avons décidé de nous lancer dans une aventure passionnante et complexe : la création d'un jeu vidéo. Le développement de jeux vidéo est un domaine en perpétuelle évolution, où la technologie et la créativité doivent cohabiter harmonieusement. Pour ce projet, nous avons décidé de mettre à l'épreuve nos compétences techniques et notre ingéniosité, tout en rendant hommage à un classique intemporel du jeu vidéo : Mario Bros. Toutefois, notre aventure ne se focalisera pas sur le célèbre plombier moustachu, mais sur son rival tout aussi emblématique, Wario. Le choix de ce projet n'est pas anodin. D'une part, il nous permet d’explorer et de maîtriser les technologies modernes en matière de développement de jeux vidéo, notamment avec l'utilisation de Unreal Engine 5.3, un moteur de jeu puissant et versatile qui offre des possibilités quasi infinies pour la création de mondes virtuels. D'autre part, en se concentrant sur Wario, nous avons la possibilité de réinventer des concepts bien établis et de proposer une vision différente de l'univers Mario, en mettant en avant un personnage souvent relégué au second plan. Cette approche nous permet également de travailler avec des éléments familiers tout en y apportant une originalité et une fraîcheur uniques. Le processus de développement de Wario Bros nous offre l'occasion de collaborer étroitement, chacun apportant ses compétences spécifiques pour atteindre un objectif commun. Lucas JOURNOUD sera en charge de la partie technique du projet, incluant la création du projet sous Unreal Engine et la gestion des aspects fonctionnels tels que les mouvements, les actions, et les interactions du personnage. Julien COSTA, quant à lui, s'occupera de la partie graphique, travaillant sur les sprites, les animations 2D, et la conception des niveaux. Cette répartition des tâches nous permettra de maximiser notre efficacité et de tirer parti de nos points forts respectifs. Ce projet est une opportunité pour nous de nous confronter à des défis réels de développement, résolvant des problèmes complexes tout en respectant des contraintes de temps. Il nous oblige à faire des choix techniques judicieux et à adopter des méthodologies de travail structurées pour assurer une progression harmonieuse et efficace. En fin de compte, notre ambition est de créer un jeu amusant et engageant, qui saura captiver les joueurs et rendre justice au riche héritage de l'univers Mario.

Table des matières

[Introduction 2](#_Toc166980849)

[Synopsis 4](#_Toc166980850)

[Méthodologie 5](#_Toc166980851)

[Répartition des tâches 5](#_Toc166980852)

[JOURNOUD 5](#_Toc166980853)

[COSTA 5](#_Toc166980854)

[Logiciels utilisés 5](#_Toc166980855)

[Choix techniques 5](#_Toc166980856)

[Développement 6](#_Toc166980857)

[JOURNOUD Lucas 6](#_Toc166980858)

[Création du Projet 6](#_Toc166980859)

[Convention 6](#_Toc166980860)

[Commande de jeu 7](#_Toc166980861)

[Création du personnage 8](#_Toc166980862)

[Création des blocks 9](#_Toc166980863)

[COSTA Julien 10](#_Toc166980864)

[Conclusions 14](#_Toc166980865)

[Annexes 15](#_Toc166980866)

[Annexe 1 15](#_Toc166980867)

[Annexe 2 16](#_Toc166980868)

# Synopsis

Dans Wario Bros, les joueurs sont invités à entrer dans la peau de Wario, l'anti-héros par excellence de l'univers Mario. Lassé de vivre dans l'ombre de Mario, Wario est déterminé à prouver sa valeur et à bâtir son propre empire, même si cela signifie d'utiliser des méthodes peu orthodoxes. Contrairement à Mario, connu pour sa bravoure et son sens de la justice, Wario préfère les solutions musclées et les ruses habiles pour atteindre ses objectifs.

L'histoire commence dans le paisible village de Wario, où ce dernier vit une existence tranquille mais insatisfaite. Toujours avide de richesses et de pouvoir, il rêve de surpasser Mario et de créer un royaume à son image. Sa chance se présente lorsqu'il découvre un portail mystique menant à divers mondes remplis de trésors et de dangers. Décidé à saisir cette opportunité, Wario se lance dans une quête épique pour accumuler le maximum de pièces d'or et de ressources nécessaires à la construction de son empire.

Au fil de son aventure, Wario traverse des mondes variés, chacun rempli de dangers, de défis, et de créatures hostiles. Il devra affronter et vaincre des ennemis féroces, résoudre des énigmes complexes, et éviter des pièges ingénieux. Ses compétences uniques lui permettront de surmonter ces obstacles : une force surhumaine pour écraser ses ennemis, des courses rapides pour esquiver les attaques, et des boules de feu dévastatrices pour détruire ses adversaires.

Chaque niveau est conçu pour offrir une expérience immersive et stimulante, où la ruse et la stratégie sont aussi importantes que la force brute. Wario devra utiliser son intelligence pour naviguer à travers des labyrinthes complexes, trouver des passages secrets, et choisir les bons moments pour attaquer ou se retirer. Les joueurs seront récompensés par des power-ups et des bonus qui amélioreront les capacités de Wario, le transformant en une force de la nature prête à défier toute opposition.

Alors que Wario progresse, il accumule des pièces d'or et des trésors, tout en bâtissant progressivement son empire. Son parcours le mènera inévitablement à une confrontation finale avec Mario, où il devra prouver une fois pour toutes qu'il n'est pas seulement une brute, mais un véritable rival digne de respect. Avec un mélange de nostalgie et de nouveautés, Wario Bros promet une aventure palpitante remplie d'adrénaline, de stratégies et de découvertes, offrant aux joueurs une nouvelle perspective sur l'univers bien-aimé des jeux Mario.

# 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

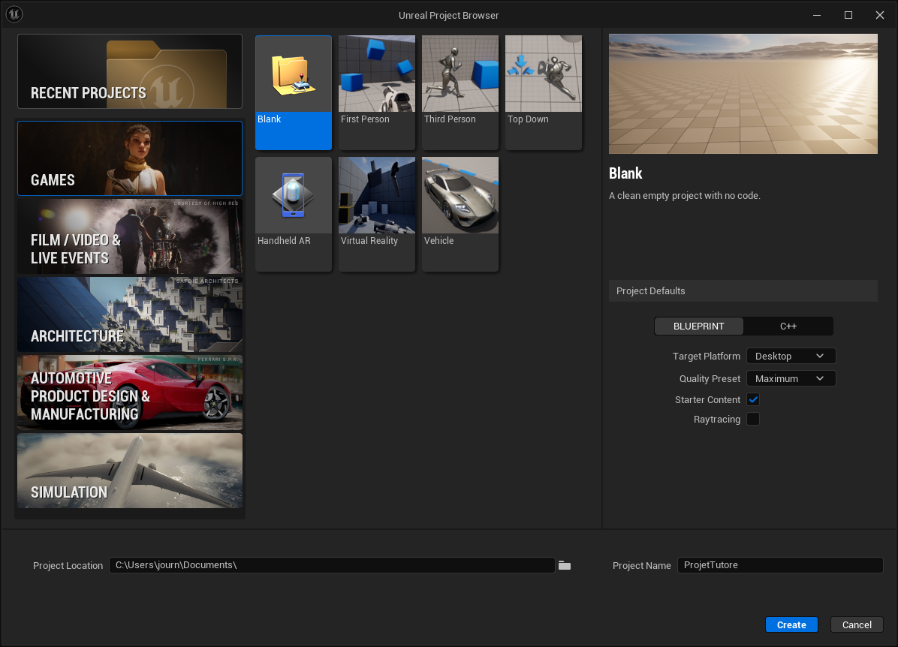
Lors de la réalisation de ce projet nous avons eu à faire plusieurs choix techniques. Pour commencer, nous avons décidé d’utiliser Unreal Engine 5.3 étant la toute dernière version disponible d’Unreal Engine a la création du projet. Cela afin de commencer notre apprentissage sur des outils à jour et 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

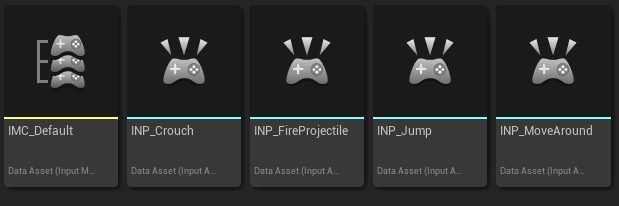
Les 2 peuvent être utiliser pour la création du même jeu indépendamment du choix fait à la création, mais Unreal Engine créera les différentes structures de fichier approprié par rapport a ce choix initial. Je voulais initialement choisir le C++, étant mon langage préféré et le maitrisant plutôt bien 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 »

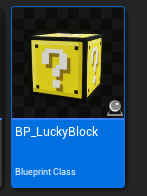
Cette fonction « Throw Fireball » permet au joueur de lancer des boules de feu. Pour commencer, on vérifie que l’évolution du joueur est égale à 2. En effet j’ai mis en place une variable évolution. Wario possédant différente forme : « petit » (valeur 0), « grand » (valeur 1) et « Wario de feu » (valeur 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 là-dessus car dans l’état actuel le joueur pourrait 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.

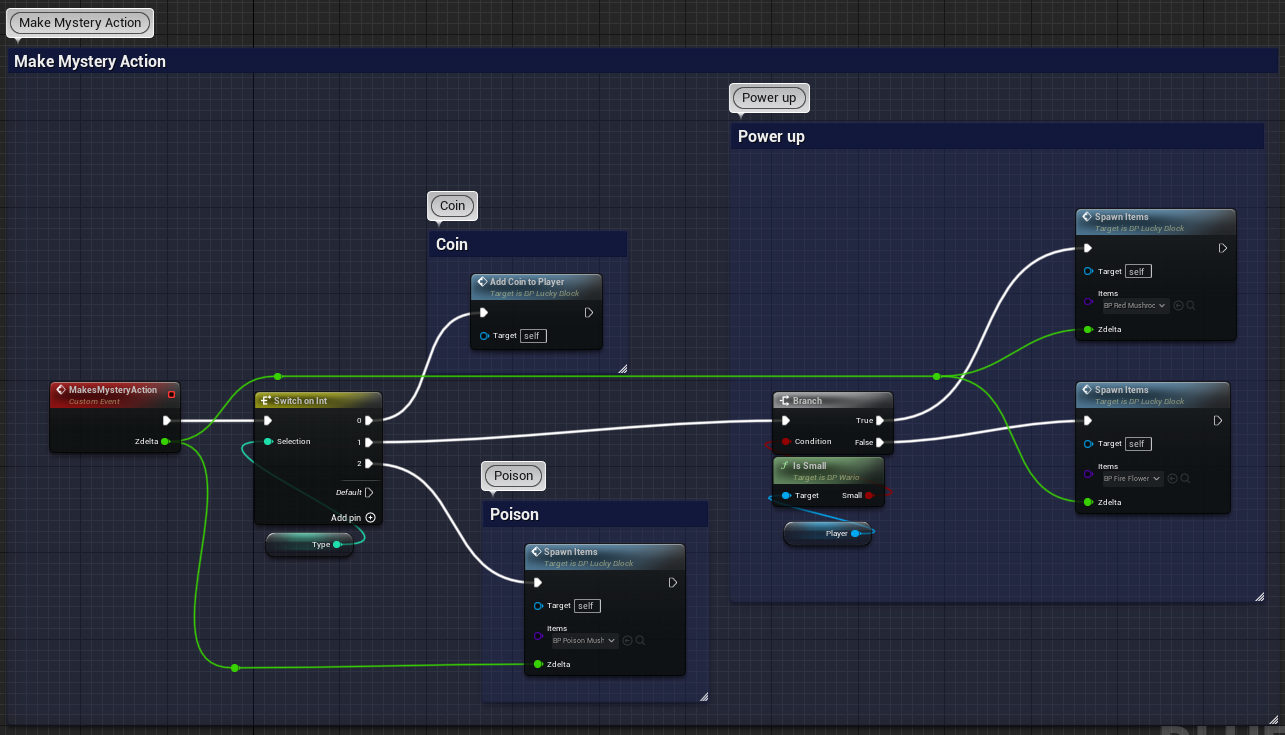
#### Contenu du block

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.



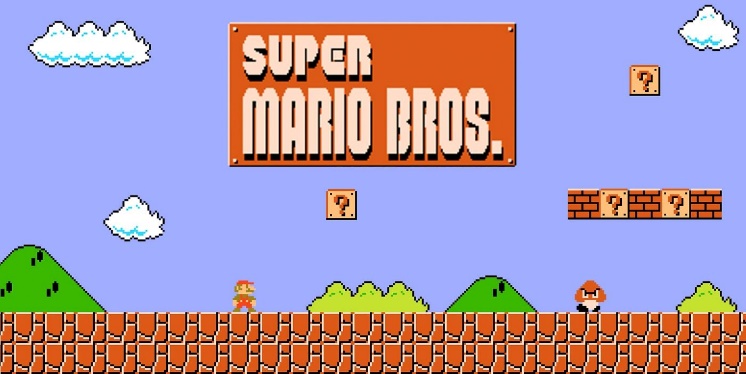
Quand cette fonction est appelée cela vérifie la valeur de la variable Type dont nous parlions avant. Si celle-ci est égale a 0, alors on appelle la fonction permettant d’ajouter 1 pièce d’or a Wario. Si elle est égale a 2 on vérifie l’évolution de Wario. Si c’est petit Wario on appelle la fonction permettant 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.

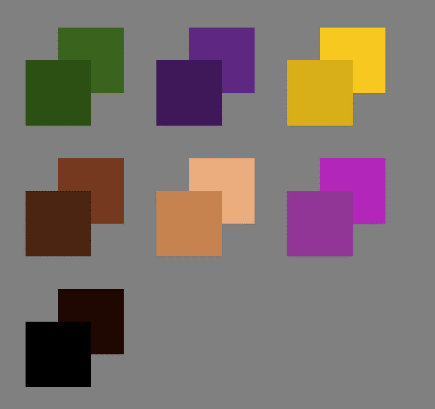
## COSTA Julien

Je me suis occupé de toute la partie graphique du projet cet à dire de la conception des sprites, de l’animation 2D sous forme de flipbook, de la création de matériaux et la création du niveau.

### Moodboard

Pour commencer j’ai d’abord fait un travail de recherche en reprenant les bases de super Mario bros concernant le sprite du personnage et les différents éléments qui le compose et j’ai repris les couleurs de wario et son univers « d’anti-héro » pour remanier les couleurs des différents ennemis.



Palette que j’ai crée pour le personnage Wario :

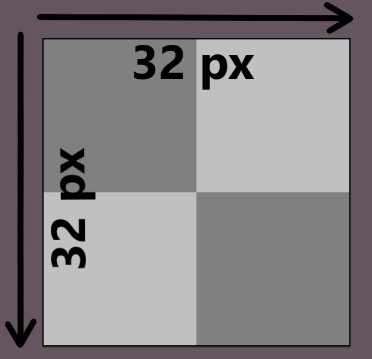
J’ai décomposé les couleurs de chaque éléments qui constitue le personnage avec la couleur principale et une couleur plus foncé qui définie son ombre.

### Création des Sprites

Pour la création des différents Sprites présent dans le jeu j’ai utilisé un logiciel payant :

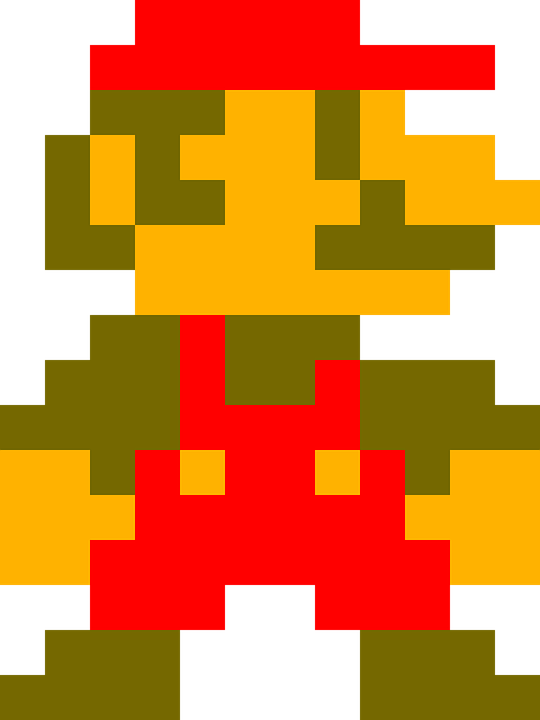


Aseprite est un logiciel de pixel-art permettant de créer des animations 2D, des sprites et tout type de graphiques pour les jeux.



Avant de crée les sprites il fallait définir une taille fixe du canva pour que tous les sprites aient la même taille de rendu et pour qu’il n’y est pas de conflit dans les tailles au niveau de Unreal Engine. J’ai donc cherché une taille idéale et j’ai vus que la plupart des jeux en pixel art utilisais le 32x32pixels alors je suis parti là-dessus comme taille par défaut de chaque canva.

J’ai alors créé un Wario en m’inspirant du Mario de Super Mario Bros et en utilisant la palette de couleur que j’avais créé au préalable. J’ai mis ci-dessous les différentes étapes tout d’abord le Mario de base du jeu puis la version en Wario qui est une version un peu remodeler du Mario en changeant les couleurs ainsi que la forme a quelques endroits. Le dernier sprite est la version finale et présente dans le jeu, celle-ci inclus les différentes ombres pour que le Wario fasse plus actuel et moins jeu rétro.

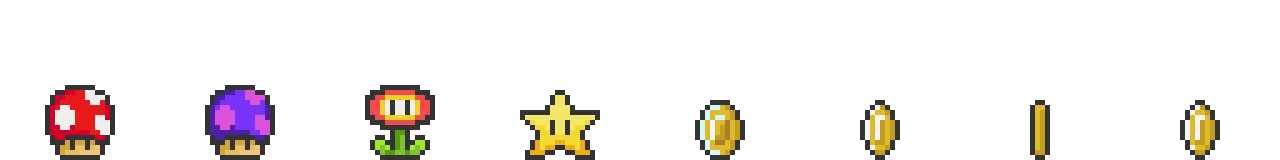
  

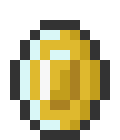
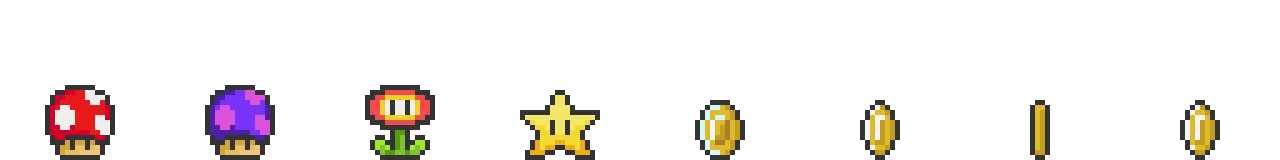


Après avoir créé la première version j’ai dû crée tous les sprites d’interactions possible du personnage c’est à dire le fait de marcher, être à l’arrêt, sauter, s’accroupir, écraser, mourir. J’ai aussi crée les animations « frame par frame » soit par exemple le fait de courir se décompose en 3 frames donc 3 sprites qui constitut le mouvement du fait de marcher ce qui donne ceci :

#### Pouvoir et collectables

Ensuite j’ai créé les différents pouvoir et collectables que Wario rencontrera durant le niveau :





J’ai mis ci-dessus la décomposition des différents sprites de la pièce puis son animation pour montrer comment fonctionne cet aspect d’animation frame par frame qui sera ensuite importer sur Unreal Engine sous forme de Flipbook.

#### Transformations de Wario

Ensuite j’ai créé les transformations de Wario en grand et en boule de feu ainsi que toutes les animations d’interactions pour chaque transformation. Ici nous allons voir l’importance d’avoir crée des sprites en taille 32x32 pixels car Wario gagne en taille et le fait d’avoir pris une taille assez grande de base permet alors d’avoir une taille conventionnelle à chaque sprites.



On se retrouve alors avec 3 Wario différents le premier, Wario « small » qui est la version de base au commencement du jeu :



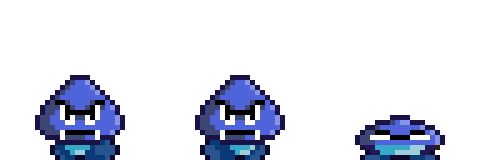
Le deuxième, Wario « tall » qui est la version améliorer par un champignon :



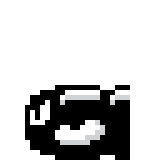
La troisième, Wario « fireball » qui est la version améliorer par une fleur de feu :



#### Ennemis

J’ai aussi créé différents ennemis et leurs animations de mouvements et interactions avec Wario en reprenant toujours l’inspiration sur le jeu Super Mario Bros mais en gardant l’esprit de Wario donc sur une palette de couleur plus froide pour montrer l’opposition avec Mario.





#### Blocs

Et pour finir j’ai fait tous les blocs qui constitue le jeu comme le bloc mystère qui donne un pouvoir ou des pièces, un bloc de brique cassable par Wario quand il est en forme grande et un bloc incassable.

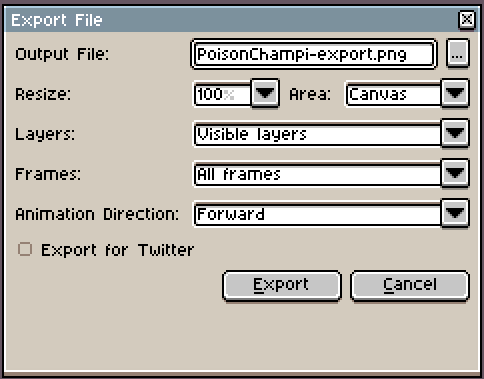
  

Ces blocs-là bien qu’ils soient des sprites vont être utiliser comme texture pour les appliqués aux différents blocs créé sur l’éditeur Unreal Engine.

### Importation vers Unreal Engine

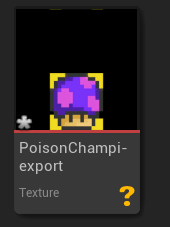
#### Import

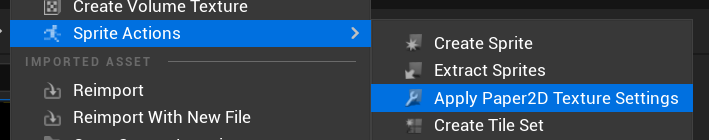
Quand les sprites sont enregistrer sur Aseprite ils possèdent l’extension propre au logiciel qui est le « .ase » pour cela j’exporte alors mes sprites en .ase depuis le logiciel en .png pour qu’il puisse garder la transparence présente dans les différents sprites.

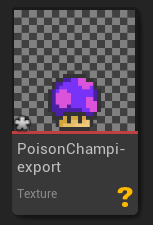


On peut aussi Resize la taille de notre png pour qu’il puisse être plus lisible visuellement depuis un logiciel permettant d’ouvrir des png mais dans notre cas on veut le garder en 32x32 pour l’importation donc on laisse a 100% de sa taille soit la taille présente sur Aseprite.

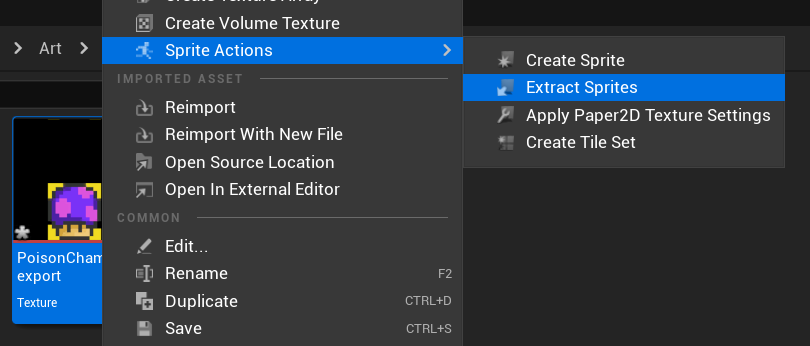
Après avoir créé les .png des différents sprites nous avons plus qu’a les importer sur Unreal Engine et leur appliquer quelques paramètres pour qu’il puisse être identique a ceux présents sur Aseprite.

A l’importation le sprite apparaît en texture sous cette forme et sans transparence du png (partie jaune sur l’image) pour remédier a ça on lui applique une texture en papier 2D.

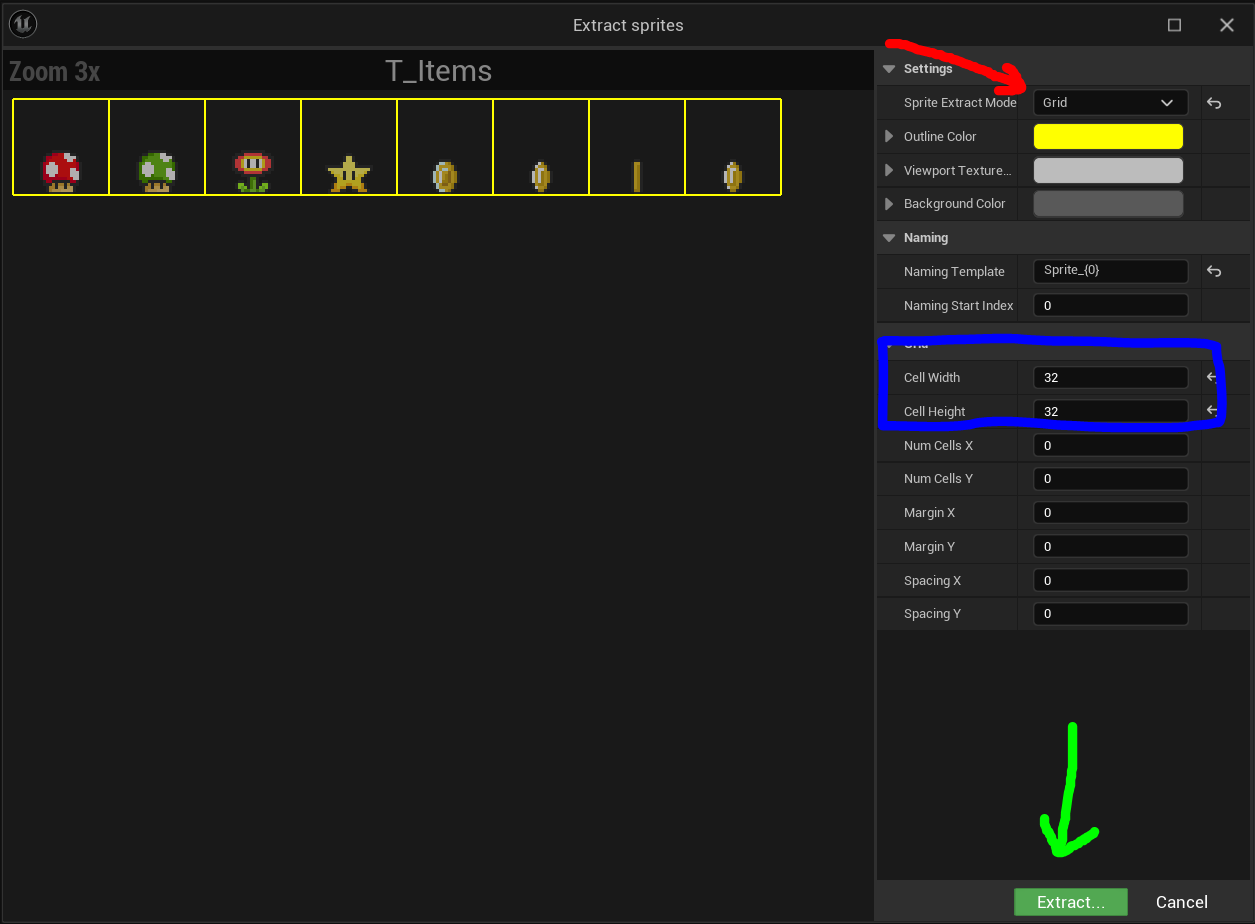




Une fois la texture en papier 2D appliqué la texture perd le fond jaune et un fond de transparence apparaît. On peut alors extraire de la texture les sprites contenu.

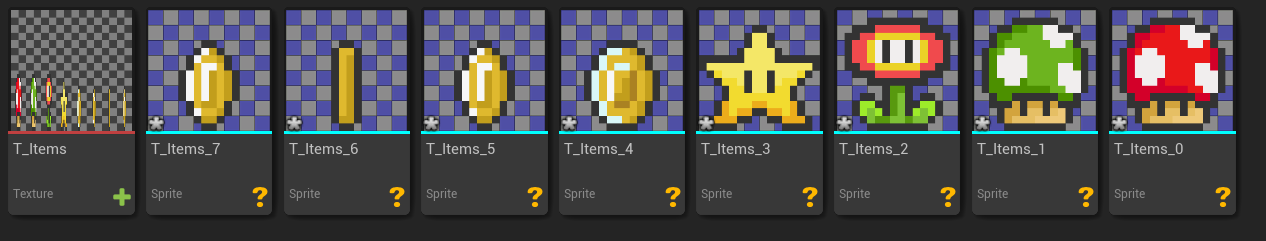


Au moment de l’extraction des sprites une interface s’ouvre et indique la découpe qu’on souhaite faire sur cette texture. Ici j’ai pris un exemple avec plusieurs sprites sur la même texture pour montrer comment faire une découpe.



Tout d’abord au niveau de la flèche rouge on enlève le mode automatique pour mettre en place le mode grid (grille), plus bas dans l’encadrer bleu on fixe la taille a 32 et 32 car nos sprites font tous 32x32 pixels soit on crée une case qui encadre le sprite tout les 32pixels en hauteur et en largeur. Enfin après avoir mis en place tout ces paramètres nous n’avons plus qu’a extraire au niveau de la flèche verte.

On se retrouve alors avec ceci :



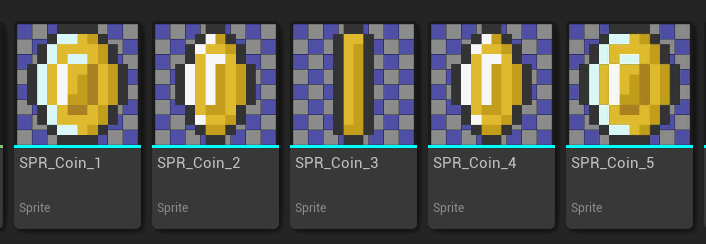
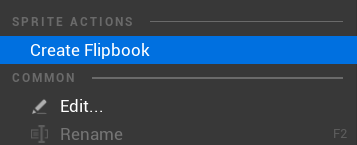
#### Convention

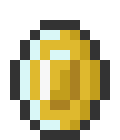
Le nomage des fichier pour ma partie se décompose en 3 préfixe :

* T\_exemple
* SPR\_exemple
* PFB\_exemple

« T\_ » pour les Textures / « SPR\_ » pour les Sprites / « PFB\_ » pour les PaperFlipBooks

#### Flipbooks

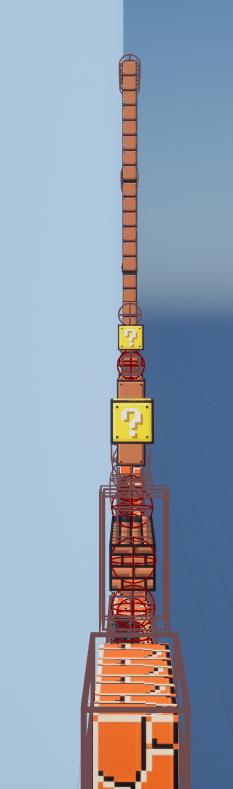
Après l’extraction des sprites et le renomage de ceux-ci nous devont crée des Flipbook qui constitut l’animation de plusieurs sprites mis a la suite avec une vitesse donnée. Il suffit alors de sélectionné les élements, de faire clic droit et create Flipbook.



On obtient alors un flipbook qui possède cette animation :

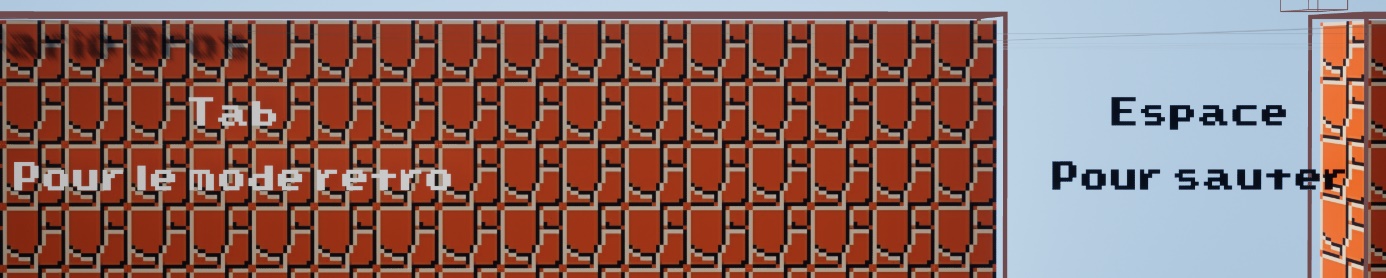
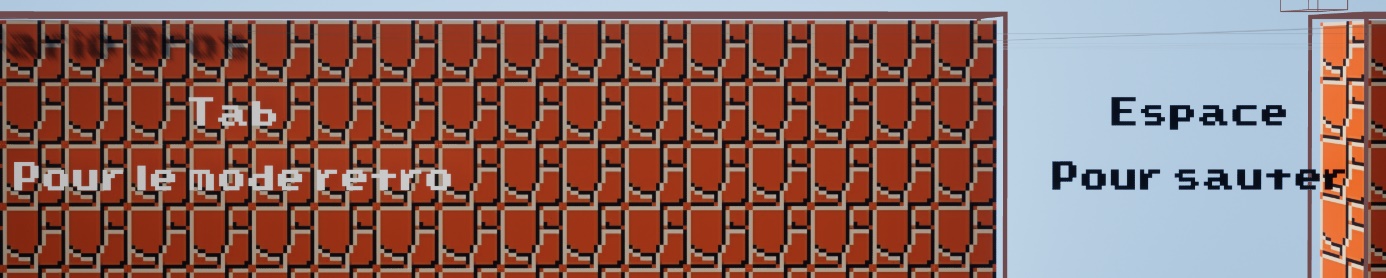
### Création du niveau

Après que toutes les fonctionnalités aient été crée ainsi que les blocs et les différentes interactions, nous avons pu créer le niveau. Celui-ci se constitut par une suite d’étapes à franchir pour arriver jusqu’à la fin du niveau. Il est assez complexe et il faut redoubler de vigilance comparée à un jeu Mario classique.

Le niveau est en 2D ce qui fait que tous les blocs doivent être posée sur un seul et même axe en ligne droite.

Nous avons aussi rajouté des indications pour aider le joueur à travers le niveau.





Nous avons rajouté différents éléments de décors pour que le jeu soit plus esthétique.



# Conclusions

Le développement de Wario Bros a été une aventure enrichissante et formatrice, nous permettant de plonger au cœur de la création de jeux vidéo. Ce projet nous a offert l'opportunité de mettre en pratique nos compétences techniques et créatives dans un contexte réel, nous préparant ainsi aux défis que nous rencontrerons dans nos futures carrières professionnelles. À travers ce projet, nous avons su exploiter les puissantes fonctionnalités de l’Unreal Engine 5.3, tout en interprétant de manière innovante l'univers emblématique de Mario Bros.

Lucas JOURNOUD a brillamment pris en main la partie technique, assurant la gestion des mouvements, des interactions, et des fonctionnalités du jeu. En optant pour les Blueprints, il a su relever le défi de s'adapter à un langage de script visuel, ce qui lui a permis de diversifier ses compétences et de s'ouvrir à de nouvelles méthodologies de développement. Julien COSTA, quant à lui, a su captiver l’essence graphique de l'univers Mario tout en apportant une touche unique à travers la conception des Sprites, des animations 2D, et des niveaux.

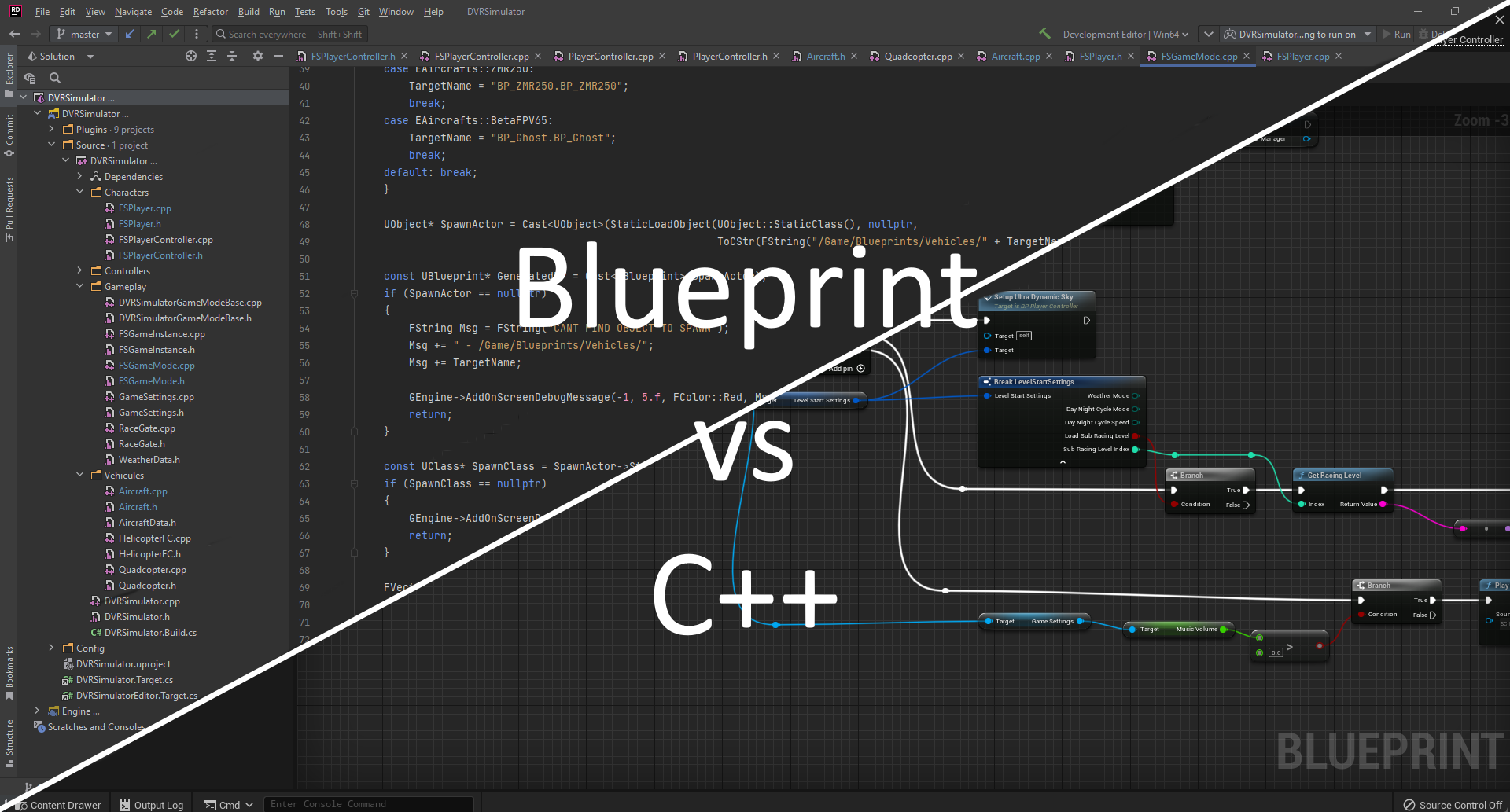
Au-delà de la simple réalisation technique, ce projet a été un exercice de collaboration et de gestion de projet. La répartition claire des tâches, le respect des conventions de nommage et d’organisation, ainsi que l’intégration fluide de nos différentes contributions ont été des éléments cruciaux pour le succès de notre initiative. Malgré les contraintes de temps et les défis imprévus, nous avons su faire preuve de résilience et d’ingéniosité pour mener à bien notre projet.

Wario Bros n'est pas seulement une réinterprétation de la franchise Mario, c’est également une déclaration de notre passion pour le jeu vidéo et de notre capacité à créer des expériences interactives captivantes. Nous avons réussi à combiner nostalgie et innovation, offrant aux joueurs une aventure dynamique et immersive où Wario, armé de sa force et de ses ruses, traverse des mondes remplis de défis et de trésors.

Enfin, ce projet nous a rappelé l'importance de l'adaptation et de l'apprentissage continu. Que ce soit par le choix des outils modernes, l'optimisation des processus de développement ou la résolution de problèmes techniques, \*\*Wario Bros\*\* s'est révélé être un laboratoire d’expérimentation et de perfectionnement pour chacun d'entre nous. Nous sommes fiers du travail accompli et enthousiastes à l'idée des futurs projets que nous aurons l’occasion de réaliser, forts de cette expérience formatrice et inspirante.

# 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)