**Rapport final sur les modifications apportées au jeu SteamBound (GameOff 2024) :**

**Introduction**

Le jeu a été fait dans le contexte du game jam GameOff2024, mais malheureusement n’a pas été complètement finalisé. Il nous restait encore prioritairement à implémenter quelques fins, à améliorer le comportement du boss et des autres AI et à ajouter des SkillBooks et d’autres livres dans le niveau.

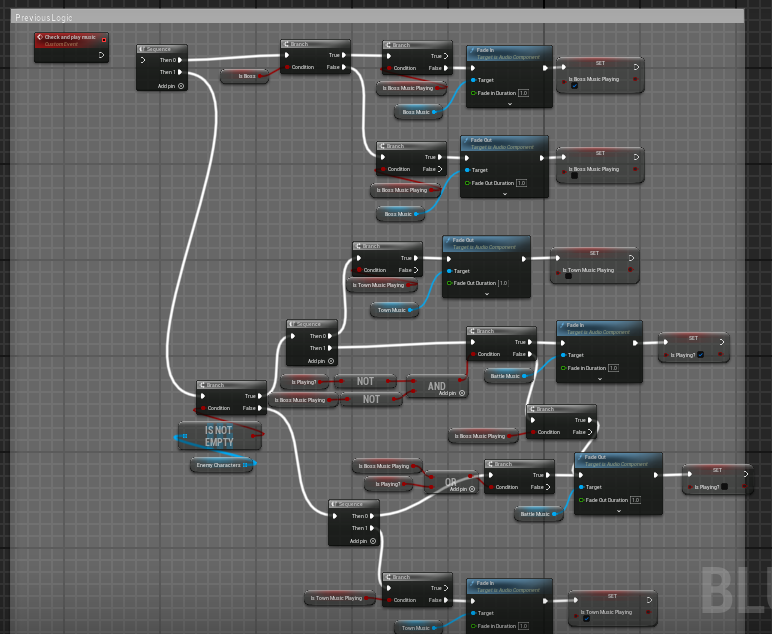
L’histoire du jeu se passe dans un petit village avec des villageois, des ennemis et le boss. Tout le monde a des pouvoirs magiques que l’on peut personnaliser en appuyant sur TAB ans le jeu. Le but est de s’amuser en explorant le village et en bien utilisant les pouvoirs pour combattre les ennemis et ainsi sauver les villageois. Les missions à faire sont affichées au fur et à mesure du jeu, mais le joueur ne peut pas les revisiter. On peut aussi interagir avec les villageois, avec des livres et d’autres objets. La durée du jeu en moyenne set de 5 à 10 minutes.

**Structure de donnée 1 : Table de hachage (HashMap dans Unreal Engine 5)**

Le blueprint BP\_MusicManager a été créé pour gérer les différentes musiques dans le jeu, en d’autres mots, à quel moment il faut changer la musique dépendamment de la présence d’ennemis. Dans la structure initiale, il y avait beaucoup de variables booléennes et plusieurs redondances de nodes.

L’ajout d’une HashMap permet d’éviter ces redondances, et permet de stocker les variables de musique dans une Map pour permettre la flexibilité et la recherche rapide de chaque musique à l’aide des Key. De plus, à la place de créer des variables pour chaque musique, on peut juste initier les sons dont on a besoin à l’aide de la node Create Sound 2D et on les ajoute à la Map.

Ancienne implémentation :

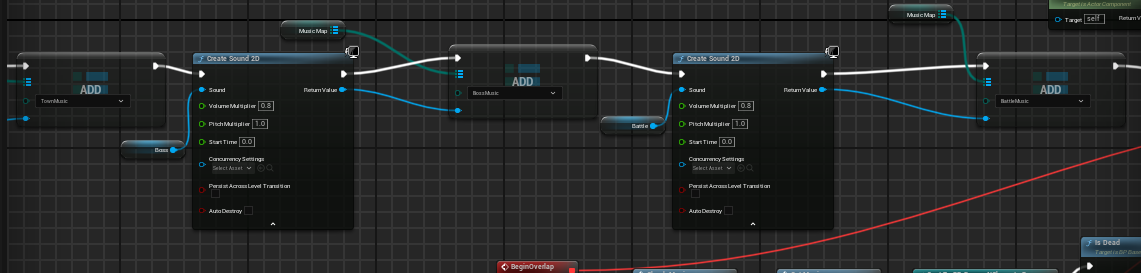


Nouvelle implémentation :

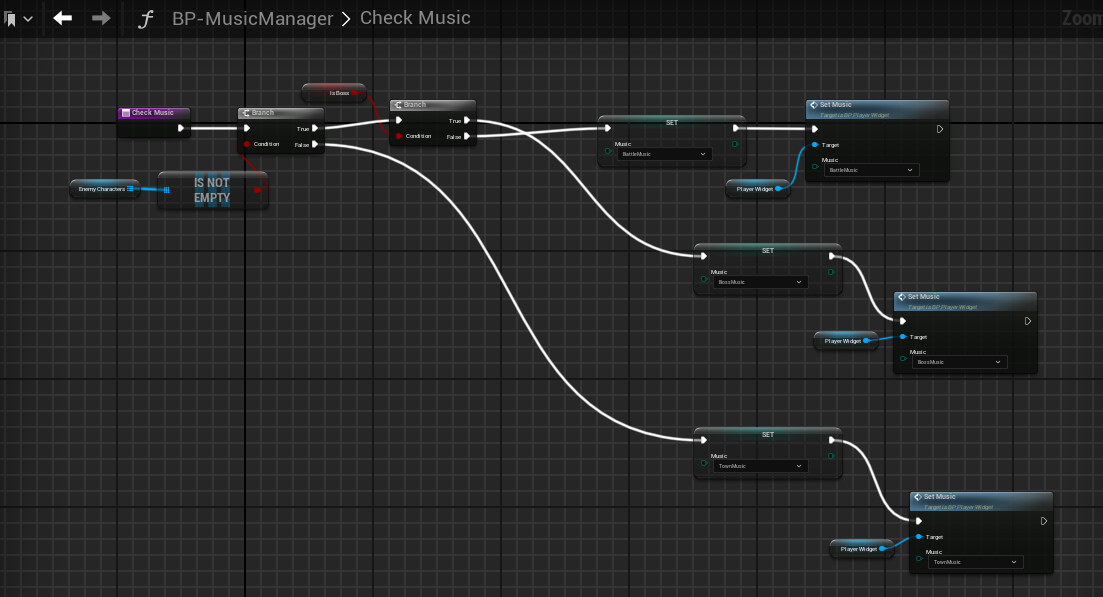
On ajoute les variables de musique à la Map

A screenshot of a computer

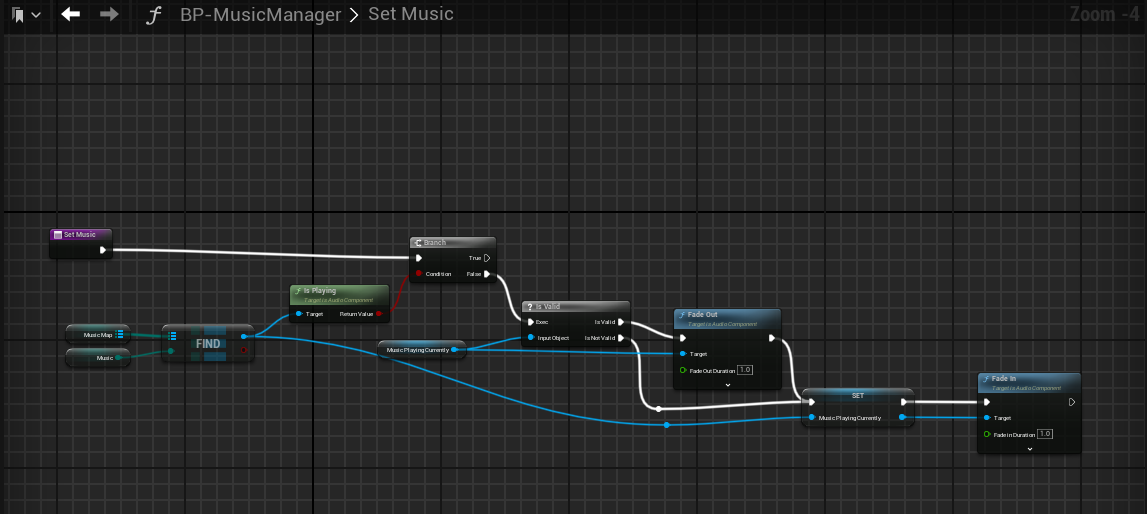
AI-generated content may be incorrect.



Fonction Check Music pour décider quelle musique doit jouer



Fonction Set Music pour changer la musique s’il le faut



Dans le jeu il y a plusieurs trigger box un peu partout, donc à chaque fois que le joueur va les passer, on va appeler les 2 fonctions pour vérifier s’il faut un changement de musique

A screenshot of a computer

AI-generated content may be incorrect.

**Algorithme : Dynamic Programming**

L’algorithme que j’ai mis en place est celui de Dynamic Programming. Il s’agit d’un algorithme d’optimisation qui permet d’améliorer un code récursif en réduisant la charge de travail faite par une fonction, ou de rassembler plusieurs blocs de code répétitifs dans une seule fonction. La façon dont j’ai appliqué cet algorithme est par la création de 2 fonctions (Set Music et Check Music) qui ont remplacé les nombreuses nodes utilisées pour créer la logique de changement de musique. À la place de créer plusieurs Branch et vérifier continuellement si telle ou telle musique joue, on peut simplement appeler ces 2 nouvelles fonctions dans le OnComponentBeginOverlap et OnComponentEndOverlap.

**Représentation visuelle :**

Pour démontrer le changement de musique tout le long du jeu, j’ai modifié le PlayerWidget en y ajoutant 2 barres verticales de chaque côté de l’écran (gauche et droite) qui vont changer de couleur dépendamment de la musique qui joue. Par défaut, la musique qui joue quand le jeu commence est la TownMusic, qui est représentée par la couleur verte. Ensuite, quand le joueur rencontre des ennemis, la couleur devient orange. Finalement, quand le joueur trouve le boss, la couleur devient rouge.

Player widget :

A screenshot of a computer

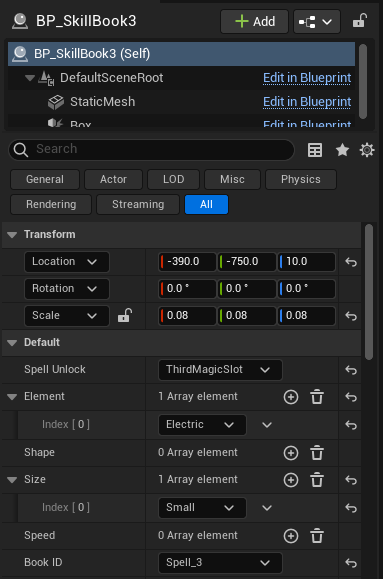
AI-generated content may be incorrect.

**Structure de donnée 2 : Tableau (Array)**

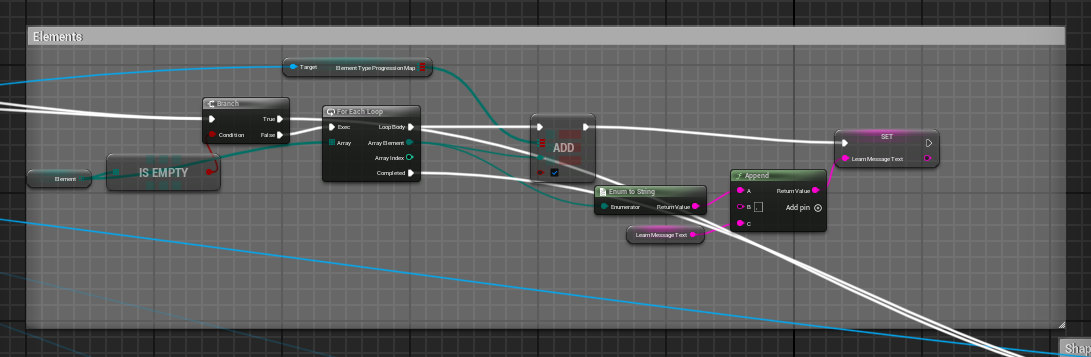
Dans le BP\_SkillBook, qui sont des livres qu’on retrouve à différents endroits dans le jeu, il y avait une logique pour faire apprendre au joueur certains « Spells » et les stocker dans une Map de progression. Quand on interagit avec un livre SkillBook dans le jeu, on peut apprendre un ou plusieurs pouvoirs, et cette logique était gérée avec une Map de progression et un seul Enum contenant les 4 Enum pour tous les pouvoirs, qu’il fallait ajouter à la Map.

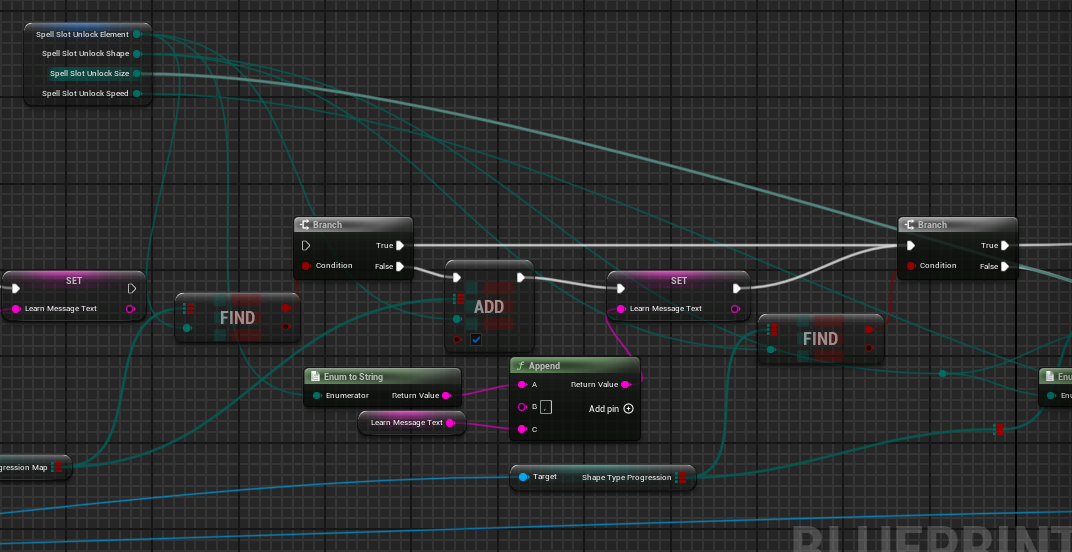
Dans le jeu, chaque SkillBook contient une description d’un seul pouvoir (par exemple le pouvoir Fast, qui permet de lancer les boules de feu avec une grande vitesse), mais notre idée était de faire apprendre au joueur plus qu’un seul pouvoir avec certains livres, même si la description s’adresse à un seul pouvoir. Pour permettre de sélectionner les pouvoirs spécifiques qu’on veut que le joueur apprenne avec chaque livre, j’ai décidé de créer 4 tableaux d’enum pour chaque type de pouvoir : Element, Shape, Size et Speed. Ainsi, dans le niveau, on peut changer les propriétés de chaque instance du Blueprint SkillBook et ajoutant les éléments dont on a besoin dans le tableau de pouvoirs correspondant.

Par exemple :



Voici la logique pour chacun des 4 tableaux de Spells :

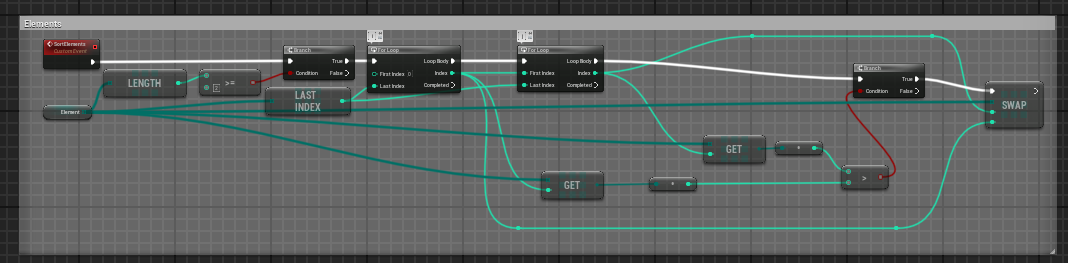
  
Avant, ça avait l’air de ça :



**Algorithme : Bubble Sort**

J’ai mis en place l’algorithme de tri Bubble Sort pour trier les pouvoirs que le joueur va acquérir lorsqu’il va interagir avec un SkillBook. Cela va permettre au joueur de visualiser les pouvoirs qu’il a appris selon l’ordre qu’il voit dans le LoadoutWidget (qu’on ouvre en appuyant sur Tab). Le Bubble Sort permet de trier les éléments dans chaque array dans l’ordre dans lequel ils apparaissent dans les enums, ainsi que dans l’ordre dans lequel ils apparaissent dans le LoadoutWidget. J’ai choisi le Bubble sort car c’est le plus facile pour moi à comprendre, et aussi car c’est facile de l’implémenter avec les nodes dans Unreal Engine.

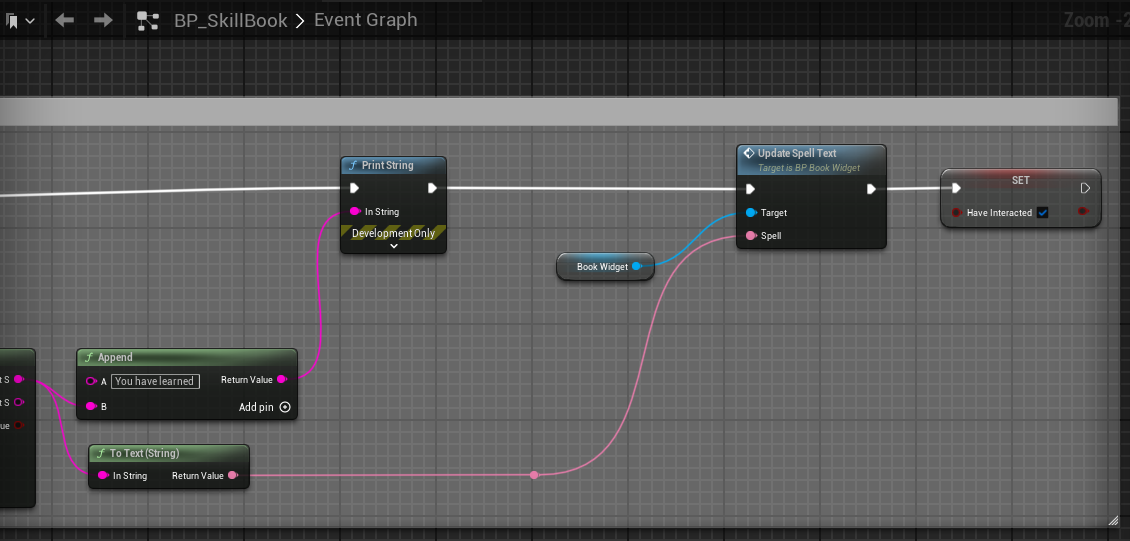
Implémentation du Bubble Sort :



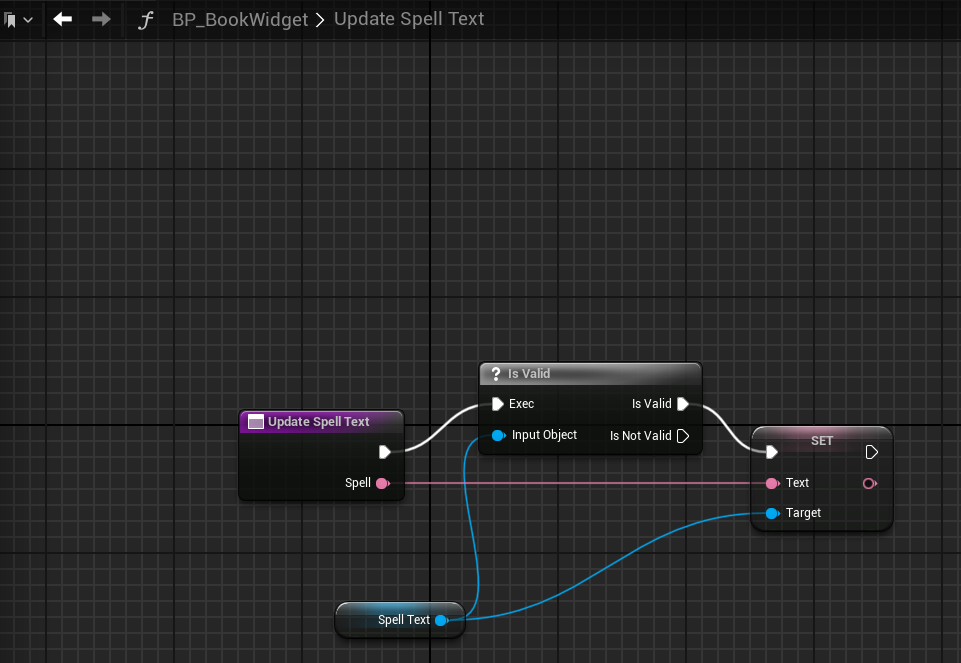
**Représentation visuelle :**

Pour rendre le jeu plus clair pour le joueur, j’ai ajouté un texte dans le BookWidget pour indiquer quel pouvoir le joueur va acquérir quand il va ouvrir un SkillBook. Originalement, les livres contenaient seulement une description du ou des pouvoirs, mais je trouve qu’on peut être plus clair et plus précis en ajoutant en bas du BookWidget le nom du ou des pouvoirs que le joueur vient d’apprendre. De plus, les pouvoirs sont triés dans ce texte (avec le bubble sort) en fonction de l’ordre par défaut qu’on peut voir en cliquant sur Tab dans le jeu.

Là où j’appelle la fonction dans SkillBook pour mettre à jour le texte dans le widget :



La fonction que j’ai créée dans le BookWidget :



**Structure de donnée 3 : Graphe (Dependency graph in Unreal Engine 5)**

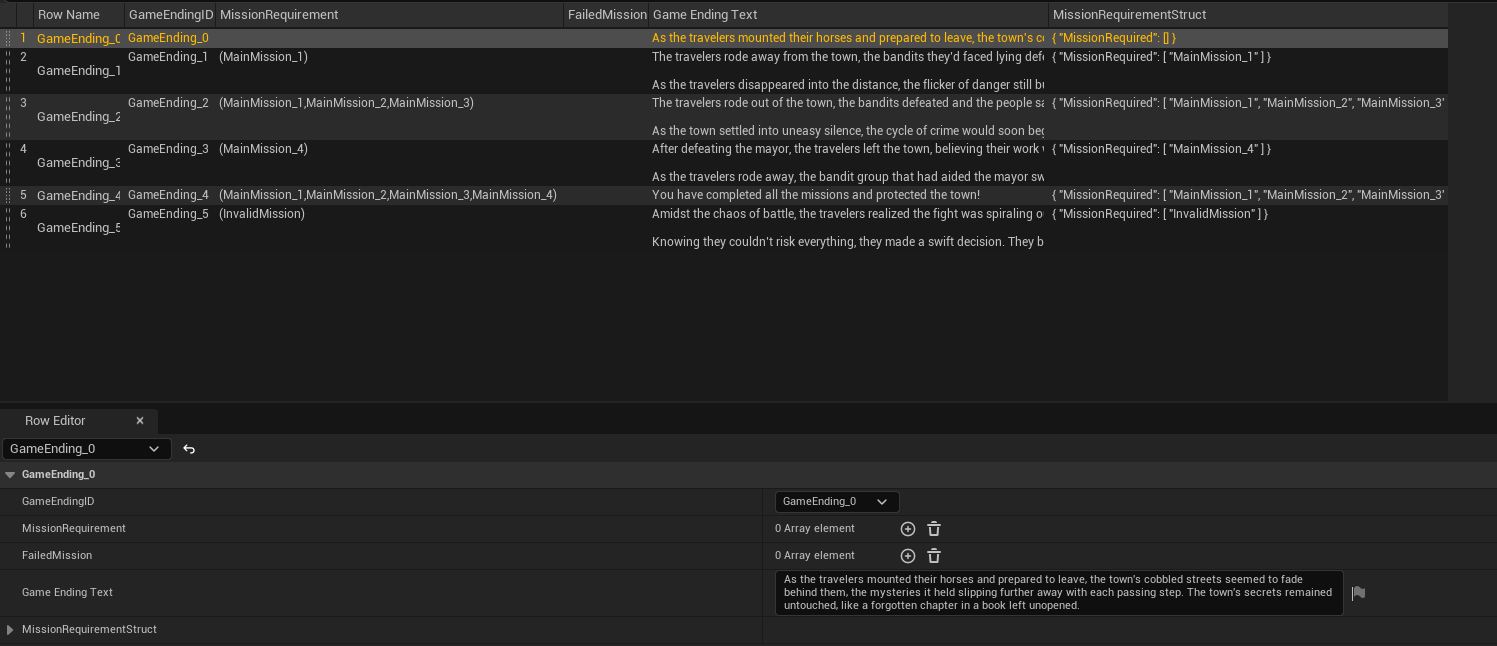
Le jeu a plusieurs fins possibles, et les conditions requises pour arriver à telle ou telle fin sont les différentes missions du jeu. Le DT\_GameEndings data table montre les missions requises pour arriver à chaque fin. Initialement, il n’y avait pas de Blueprint pour gérer les fins, et elles étaient gérées dans un Widget et le BP\_CameraController, ce que j’ai trouvé trop éparpillé. De plus, le widget ne doit pas être responsable de gérer les fins, ni le CameraController, car ce dernier doit seulement contenir les informations liées à la caméra. La fonction initiale pour évaluer la fin à afficher n’avait rien pour stocker les fins qui sont déjà débloquées, ce dont pourquoi j’ai créé 2 Maps pour stocker cette information.

**La première Map** stocke les fins et les missions requises correspondantes, utilisée pour itérer à travers les missions et vérifier si toutes les missions requises sont parmi les missions que le joueur a complétées.

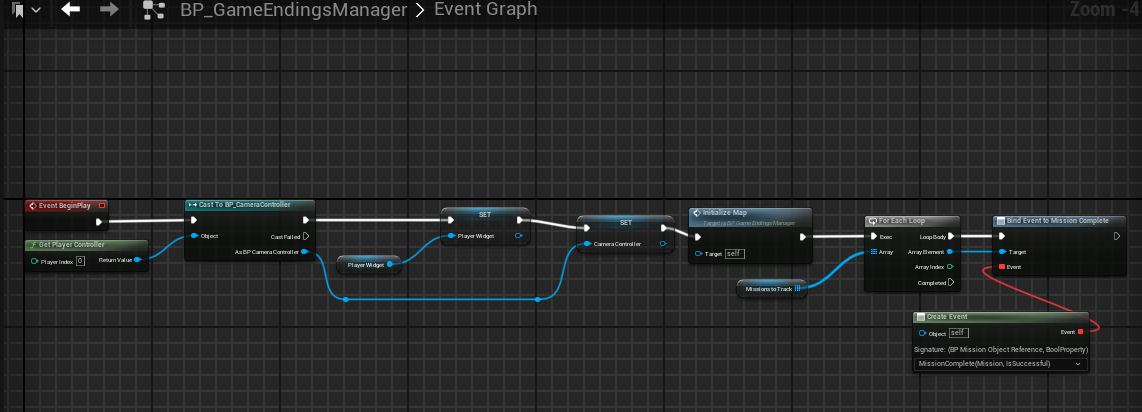
**La deuxième Map** stocke les fins avec une valeur booléenne (Unlocked). Cela sert à vérifier si certaines fins sont déjà débloquées. Si non, on va continuer avec les vérifications et quand on déloque une fin, on va l’ajouter dans la Map avec Unlocked marqué True.

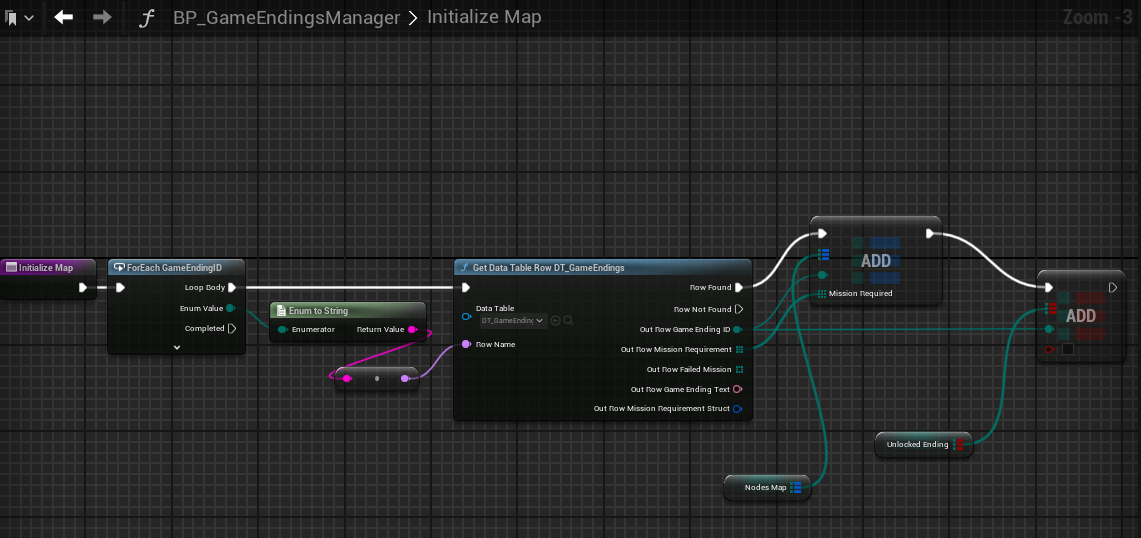
Pour correctement implémenter cet algorithme, j’ai aussi dû modifier les blueprints BP\_Mission, BP\_CameraController et le GameEnd widget.

Voici la Data Table (la fin numéro 5 n’a pas été implémentée) :



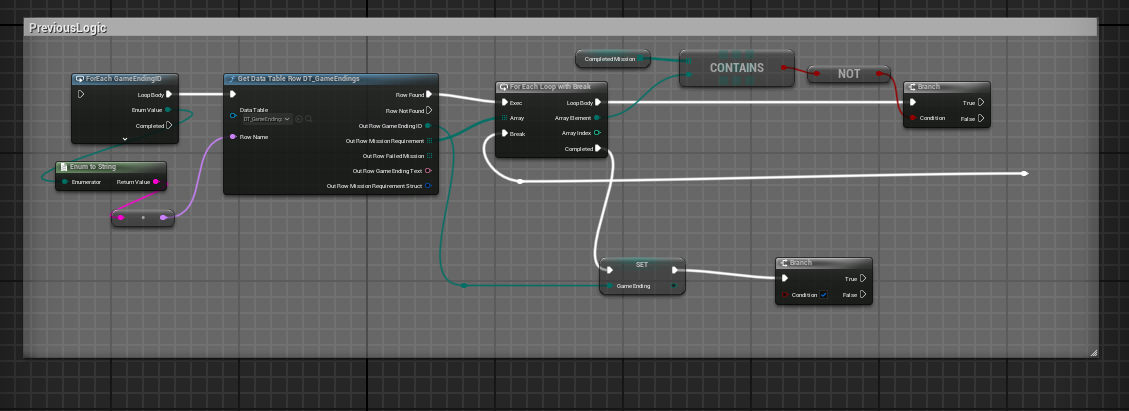
Nouveau Blueprint GameEndingsManager créé pour le graphe :





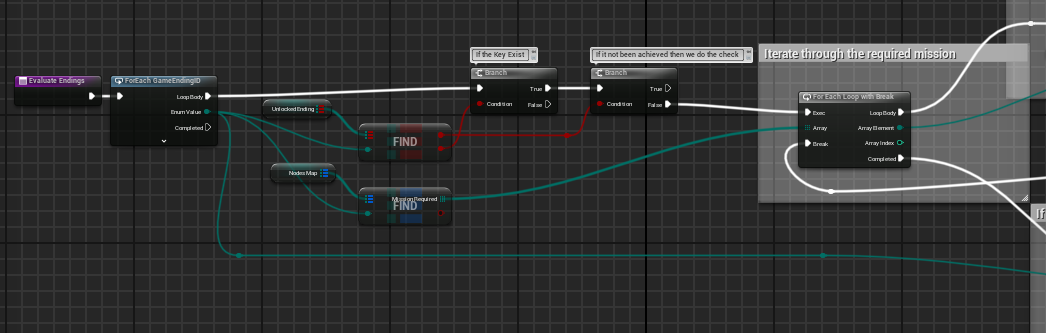
Graphe :

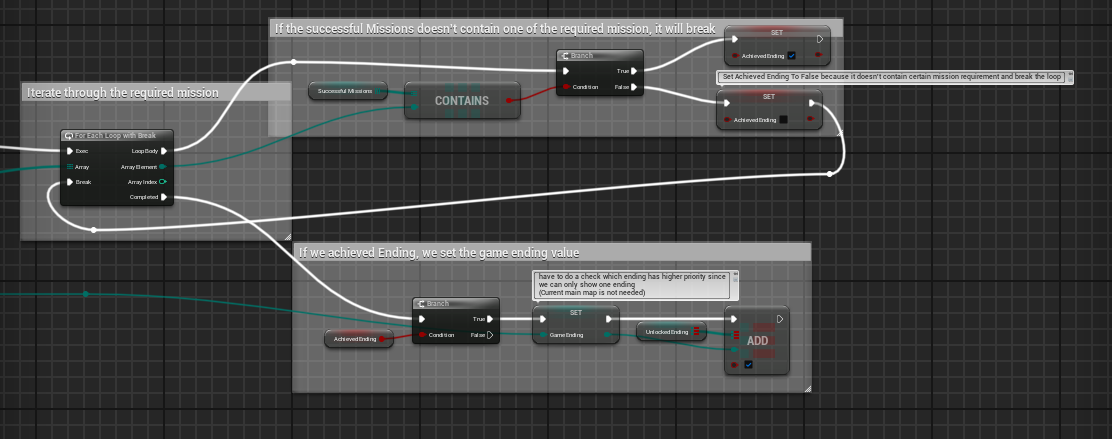
Ancienne implémentation :



Nouvelle implémentation :

La fonction EvaluateEndings est appelée dans la fonction MissionComplete (qui était déjà créée à la base), qui elle, est appelée à chaque fois que le joueur complète une mission.



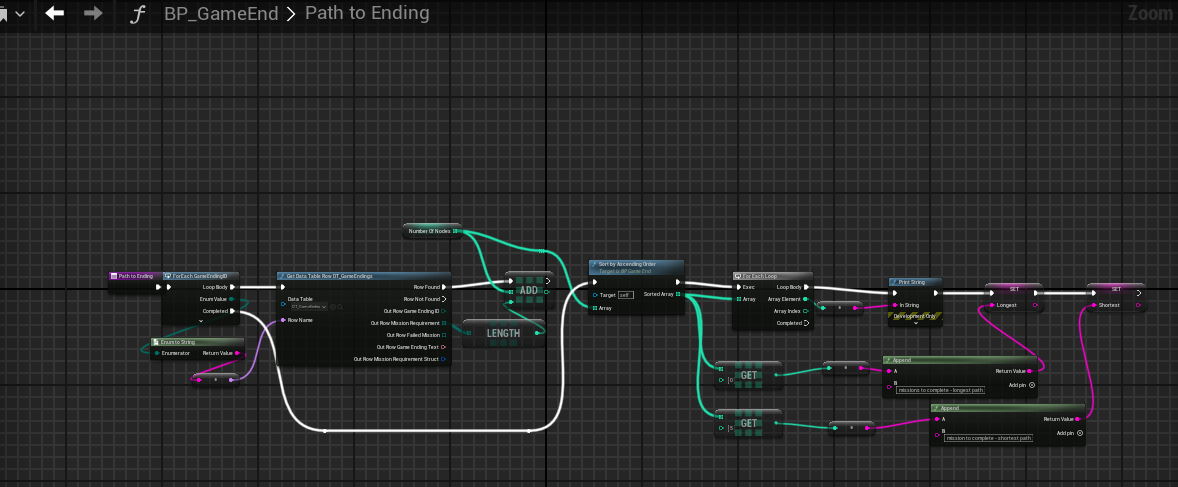


**Algorithme : Bubble sort avec une fonction personnalisée**

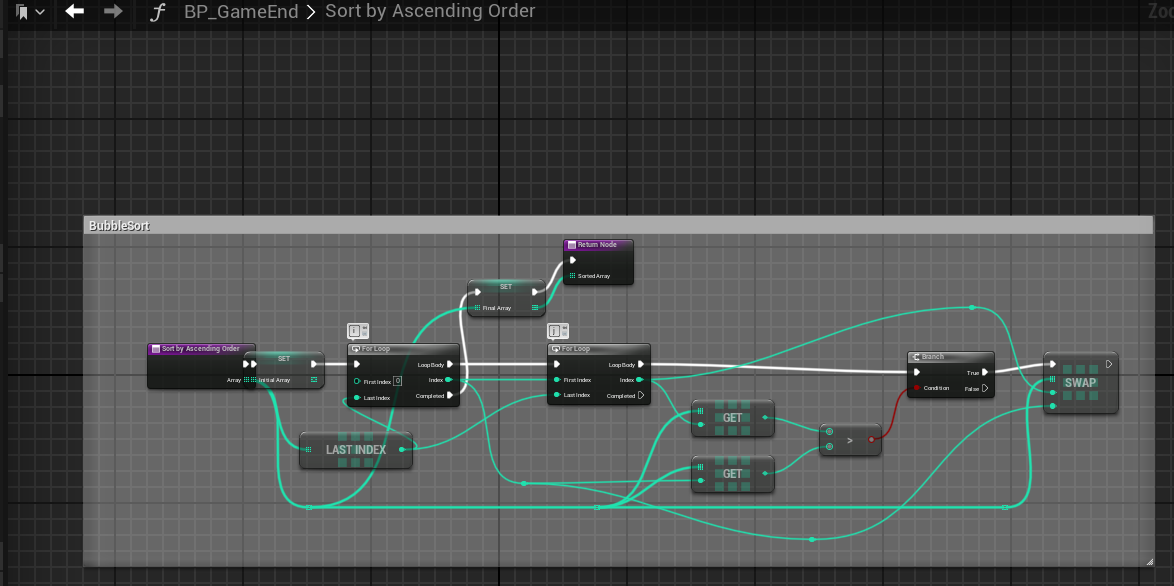
Puisque le joueur n’a aucune connaissance des fins possibles du jeu et ne peut pas voir une liste des missions à faire ou complétées, j’ai créé un algorithme qui trouve la fin la plus longue (qui requiert le plus de missions pour l’atteindre) et la fin la plus courte (qui requiert 0 mission). Quand le widget GameEnd sera affiché, le joueur pourra voir s’il a réussi à débloquer la fin la plus courte et la fin la plus longue, ainsi que l’information à propos du nombre de missions pour s’y rendre. Pour faciliter la compréhension pour les joueurs, j’ai mis la représentation visuelle de cet algorithme dans le GameEnd widget.

Le Bubble Sort trie de façon croissante les longueurs des tableaux de missions requises, et la fonction PathToEnding va prendre les valeurs triées et les assigner à 2 string, un qui montre le shortest path to ending et un qui montre le longest. Comme pour l’algorithme dans le SkillBook, j’ai juste choisi le Bubble sort car c’est le plus facile pour moi à implémenter.

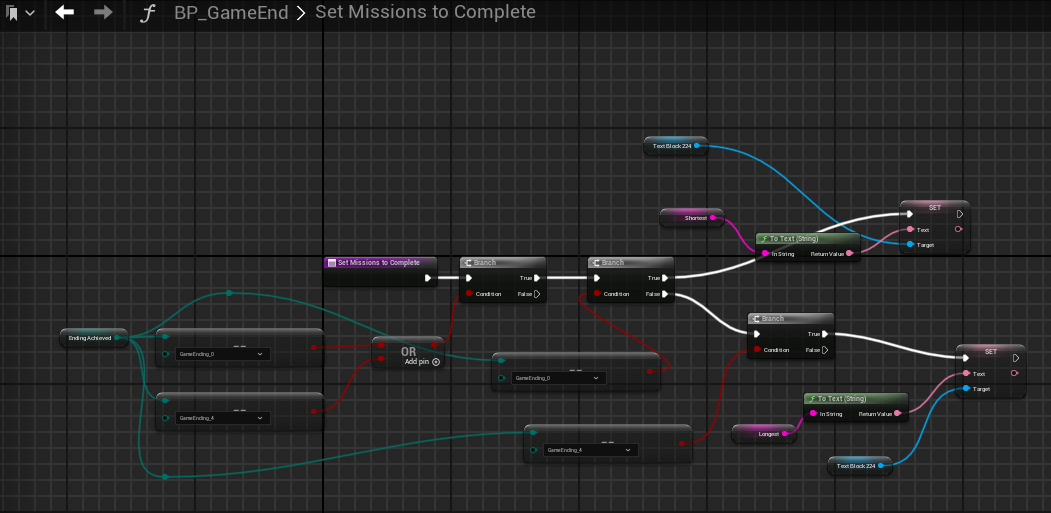
La fonction PathToEnding :



La fonction de Bubble sort :



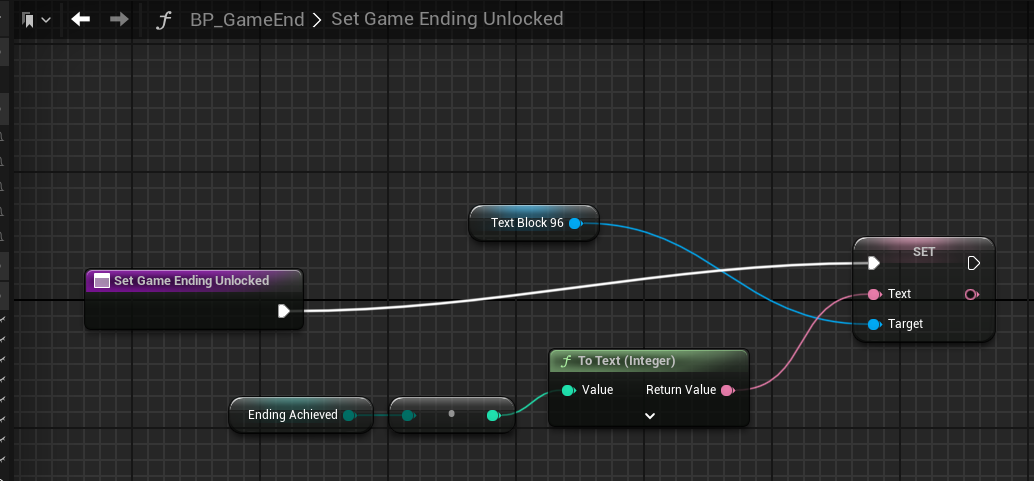
La fonction pour afficher le texte des fins (courte et longue) :



**Représentation visuelle :**

Dès que le GameEnd est triggered, c’est-à-dire dès que le joueur décide de sortir du village par les portes ou qu’il tue le boss (les 2 façons possibles de trigger le GameEnd), il va y avoir un widget qui annonce la fin du jeu et la description de la fin atteinte. Cependant, rien ne montre quelle fin exactement on vient de débloquer. J’ai donc ajouté une fonction dans le GameEnd widget qui affiche la fin atteinte (0 jusqu’à 4).

Fonction qui affiche la fin atteinte (le numéro) :



**Intégration des commentaires :**

Durant la présentation finale, vous m’aviez demandé comment de temps cela prenait pour atteindre la fin 4 (la fin qui requiert le plus grand nombre de missions). Cela m’a fait penser que je devrais réduire le nombre d’ennemis pour faciliter la tâche aux personnes qui vont tester le jeu. Par exemple, la mission 3 dans le jeu était complétée avec succès quand on réussit à tuer tous les 4 ennemis dans le grand bâtiment, mais cela prend quand même plusieurs essais pour les tuer, alors j’ai réduit le nombre d’ennemis à 1 pour des fins de testage. J’inclus aussi une vue d’en haut de la carte du jeu pour montrer les endroits de chaque mission.

A video game screen shot of a town

AI-generated content may be incorrect.

J’ai fait similairement pour les SkillBook. Puisque les SkillBook sont à différents endroits dans le jeu, cela prend du temps pour les retrouver tous, alors j’ai un peu changé leur position. Voici une capture d’écran pour les montrer (il y en a 5) :

A video game screen shot of a town

AI-generated content may be incorrect.

Le lien vers le build du jeu est disponible sur le Github du jeu : <https://github.com/KuroiRaku/ProjectGameOff2024>

**Réflexion**

Le graphe et les algorithmes de tri (surtout le Bubble sort) sont des éléments que j’ai pu beaucoup approfondir grâce à ce travail. Surtout en travaillant dans Unreal Engine avec les Blueprints, j’ai trouvé ça assez difficile d’implémenter le graphe et le Bubble sort car j’étais plus confortable à les faire en C++. Je suis tout de même contente d’avoir travaillé avec Unreal Engine pour ce projet parce que cela m’a permis d’explorer davantage la logique des Blueprints et de mettre ma créativité en jeu (surtout pour l’implémentation du graphe).

Au terme de ce cours, j’ai réalisé qu’il faut avoir quand même beaucoup d’expérience dans la création de jeux et leur codage pour savoir dans quels cas il faut appliquer telle ou telle structure de données. Comme vous nous l’aviez dit, ce n’est pas naturel au début de savoir qu’à tel endroit dans le code, une liste chaînée est plus appropriée, par exemple. Je pense donc que c’est ça qui sera le défi le plus grand à relever au niveau des structures de données et des algorithmes. Individuellement, je me débrouillais assez bien pour les comprendre et les implémenter, mais dans un vrai jeu, c’était dur de savoir à quel endroit les implémenter et surtout comment les implémenter dans telle situation ou tel cas spécifique. Pour ce projet, le graphe a définitivement été le plus grand défi pour moi car c’est la structure de donnée que j’ai le plus de misère à concevoir et imaginer dans ma tête. J’ai dû faire des petits dessins pour trouver comment faire la logique des nodes et faire de la recherche pour trouver quelques exemples concrets d’application des graphes. C’est ainsi que j’ai réussi à l’implémenter et à mieux le comprendre.

Après que j’obtiendrais mon diplôme, je pense peut-être continuer à l’université pour étudier le développement des jeux plus en profondeur. Entre-temps, je vais essayer de participer à plusieurs game jams ou d’autres activités liées aux jeux. J’ai aussi réalisé que j’ai besoin de plus pratique dans le C++, alors j’aimerais essayer de participer à des challenges de codage ou des hackathons. Dans mon temps libre, je vais aussi me pratiquer avec le AdventOfCode car j’aime bien comment les problèmes sont rédigés et ça me fait beaucoup réfléchir.

J’ai bien aimé le projet de William. Le design de son jeu est très unique (tout en blanc) et ses mécaniques de clones sont aussi très originales et intéressantes. Bien que le jeu en tant que tel est très simple, je crois qu’on pourrait y avoir beaucoup de variations de gameplay. L’idée de labyrinthe de Noémie m’a aussi très impressionnée, surtout sa mécanique de génération du labyrinthe en temps réel.

Bien que j’aie vraiment apprécié ce jeu fait dans le contexte d’un game jam, je ne suis pas sûre de vouloir le continuer dans mon temps libre. Par contre, j’ai l’intention de continuer mon jeu de Moteur de jeu 2, qui est un jeu de lasers inspiré un peu par le jeu Portal. J’ai voulu y inclure des portails, mais je n’ai pas encore réussi à faire fonctionner les portails comme dans le jeu Portal, alors je vais continuer à travailler dessus. Mon plan était aussi de créer des niveaux comme dans Portal où tu dois trouver une façon de se rendre à la sortie en plaçant les portails aux bons endroits, mais j’ai manqué de temps. Je voudrais donc continuer sur le design des niveaux et la mécanique des portails pour y faire mon propre jeu de portails unique.