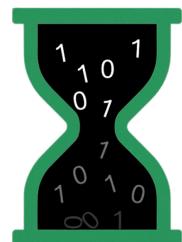


Dtres



Rapport de Projet

CodeFall



Promotion 2029

Florian GAGNIER
Imane SARWAR
Florent FOSSATI
Alexandre SALGUEIRO

C2

Table des matières

Introduction	4
I Vue générale du projet	5
1 Les membres du groupe	5
A Sarwar Imane	5
B Gagnier Florian	5
C Fossati Florent	5
D Salgueiro Alexandre	6
2 But du projet	6
3 Moyens utilisés	7
4 Répartition des tâches	8
5 Ressenti général	8
II Avancement du projet	9
1 Site Web	9
2 Scénario	11
A Contexte général et mise en situation	11
B Une narration en trois temps : de la découverte à la désillusion	11
C Dialogues, tonalité et narration indirecte	12
D Un message écologique et humain déguisé en gameplay	13
E Des mécaniques qui racontent autant que les mots	13
3 Mécaniques de jeu/Gameplay	14
A Portes	14
B Livres	15
C Dialogues	16
D Caisses	17
E Indices	19
4 Enigme	20
A Spawn	20
B Bibliothèque	20
C Labyrinthe	21
D Les salles de classes	23
5 Multijoueur	25
A Interface	27
B Déconnexion	27
C Caméras	27
D Lobby	28
E Porte	29
F Caisse	30
G Robots	31
H Animations	31
I Conclusion	31
6 Interface Utilisateur	32
A Menus	32
B Option	35
C Crédit	35

7	Intelligences artificielles	36
8	Graphismes	37
9	Animations	46
	A Personnage	46
	B Robots	47
10	Musiques et sons	48
	III Bilan sur l'avancement	50
	IV Conclusion	50

Introduction

Dans le cadre de notre projet informatique de second semestre, nous avons décidé de réaliser un jeu vidéo : **CodeFall**.

CodeFall est un jeu d'aventure et d'énigmes jouable à deux, en vue à la première personne. Le joueur incarne un humain plongé dans un monde "buggé" où la résolution d'énigmes permet d'évoluer et de mieux comprendre son environnement.

Chaque joueur est accompagné d'un petit robot qui le suit, lui parle et l'aide dans son aventure. Ces robots, Bip et Bop, expliquent tout au long du jeu l'importance des actions entreprises par le joueur. Ces actions modifient le monde et contribuent à le rendre instable.

Nous vous proposons, par rapport, de découvrir comment **CodeFall** a été conçu et réalisé par le groupe **Dtres**.

I. Vue générale du projet

1. Les membres du groupe

A. Sarwar Imane



Passionnée par le potentiel de la technologie à transformer le monde, j'ai toujours cherché à comprendre et optimiser les systèmes, mais ce qui me motive avant tout, c'est d'utiliser ces compétences pour avoir un impact positif sur la société. C'est ce qui m'a conduit à co-fonder Dtres, une entreprise de jeux vidéo engagée à sensibiliser aux grands enjeux mondiaux.

Au sein de l'équipe, j'apporte non seulement mes compétences techniques, mais aussi une vision plus humaine, philosophique, pour créer des solutions éthiques à la fois innovantes et porteuses de sens.

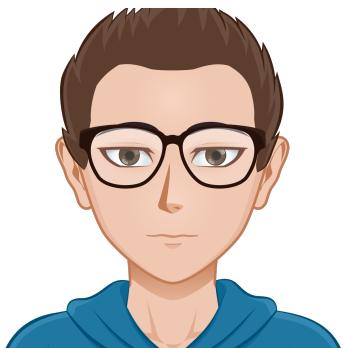
B. Gagnier Florian



Passionné de jeux vidéo depuis mon enfance, j'ai rapidement voulu passer de joueur à créateur. J'ai donc appris plusieurs langages de programmation, ce qui m'a incité à explorer différents moteurs de jeu tel que Unreal Engine et surtout d'Unity 3D. Souhaitant aller plus loin dans ce domaine, j'ai choisi de poursuivre mes études à EPITA pour approfondir mes connaissances en informatique et continuer à transformer ma passion en véritable expertise.

Ce projet est une opportunité de mettre en pratique mes connaissances, mais aussi d'en apprendre davantage et de progresser en gestion de projet et en communication.

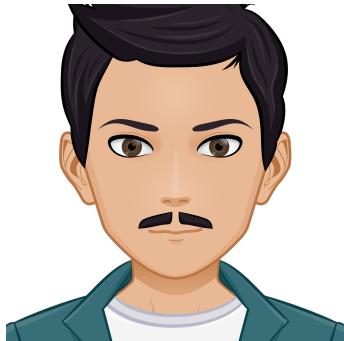
C. Fossati Florent



J'ai longtemps été joueur, mais j'ai toujours eu envie de passer de l'autre côté du jeu, de créer à mon tour également. Pour cela j'ai déjà utilisé unity pour m'initier au monde de la création de jeu. Par ailleurs, je fais de la musique depuis mon enfance, dont trois années de MAO (musique assistée par ordinateur). Cela m'a naturellement amené à prendre en charge l'ambiance sonore du jeu.

Ce projet m'a permis d'acquérir de l'expérience en plus de me montrer à quoi ressemble la création en équipe d'un jeu. Grâce à ce projet j'ai pu concrétiser l'un de mes rêves d'enfant : faire un jeu vidéo.

D. Salgueiro Alexandre



Ma passion pour les jeux vidéo remonte à l'école primaire. Depuis, j'ai exploré différents jeux et plateformes, ce qui a nourri mon intérêt pour l'informatique. En seconde, j'ai fait mes premiers pas dans ce domaine avec le cours de SNT, en découvrant Python. Cette expérience m'a incité à choisir la spécialité Sciences de l'ingénieur en première, où j'ai approfondi mes connaissances en informatique et découvert la modélisation 3D.

J'ai par la suite expérimenté Blender afin de pouvoir faire prendre vie à des images. De part cette expérience, je suis principalement responsable du design du jeu dans notre projet. Ce projet m'a non seulement permis d'approfondir mes compétences techniques mais aussi d'apprendre à travailler de manière plus efficace et d'améliorer ma capacité à travailler en équipe.

2. But du projet

L'objectif de ce projet était de nous apprendre à travailler en équipe sur un projet d'envergure, tout en améliorant nos compétences en communication, en collaboration et en gestion de projet.

La création d'un jeu vidéo mobilise de nombreux outils qu'il est essentiel de maîtriser. Cela nous a permis d'enrichir nos compétences dans divers domaines comme le game design, la programmation, l'animation, la musique et l'écriture scénaristique.

Mener à bien un projet aussi complexe et collaboratif représente également une excellente opportunité pour renforcer notre confiance en nous, d'améliorer nos relations avec autrui, apprendre à respecter des deadlines et à surmonter des obstacles techniques et créatifs.

3. Moyens utilisés



Unity : Une plateforme de développement très utilisée pour créer des jeux et expériences en 2D, 3D, VR et AR, apprécié pour sa polyvalence et sa large communauté.



JetBrains Rider : Un IDE (environnement de développement intégré) multiplateforme pour le développement .NET et C#.



Visual Studio Code : Un éditeur de code open source et léger, développé par Microsoft qui supporte de nombreux langages de programmation.



Git : Un système de contrôle de version distribué, largement utilisé par les développeurs pour gérer les versions de code source, collaborer et suivre les modifications.



GitHub : Une plateforme basée sur Git, qui permet de stocker, gérer et collaborer sur des projets de développement logiciel.



Blender : Un logiciel open source de modélisation 3D, très utilisé pour la création d'animations, de jeux et d'effets visuels.



Photon : Un fournisseur de services multijoueurs pour les jeux vidéo, offrant une plateforme de mise en réseau et de programmation pour les développeurs.



FL studio : est un logiciel d'édition et de création sonore électronique, dans notre projet nous allons l'utiliser pour faire les musiques du jeu.

4. Répartition des tâches

Répartition des tâches	Gagnier Florian	Fossati Florent	Salgueiro Alexandre	Sarwar Imane
Menu/Options			*	+
Multijoueur	*	+		
Audio		*	+	
IA		*		+
Enigmes et Mécaniques de jeu	+	*		
histoire et dialogues	+			*
Graphismes/Texture			*	+
Site Web	*		+	

Légende	
*	Responsable
+	Aide directe

5. Ressenti général

Dans l'ensemble, ce projet de création de jeu vidéo a été une expérience extrêmement enrichissante pour toute l'équipe.

Travailler ensemble sur toutes les étapes, de la conception à la réalisation, nous a permis de développer des compétences techniques (programmation, graphisme...), mais aussi des compétences humaines essentielles : communication, gestion du temps, et travail en équipe.

Nous avons rencontré des défis, notamment en termes de coordination, de bugs inattendus et de délais parfois serrés. Mais nous avons su nous adapter, persévérer, et rester motivés jusqu'au bout.

Ce projet nous a aussi permis de mieux comprendre les coulisses de la création vidéoludique et de mesurer l'importance d'une organisation réfléchie et d'une répartition claire des tâches.

Nous sommes fiers du résultat final, même si nous savons qu'il y aurait encore beaucoup de choses à peaufiner. Mais plus que le produit lui-même, c'est le chemin parcouru en équipe qui reste le plus marquant.

II. Avancement du projet

1. Site Web

Nous avons conçu et mis en ligne un site web en utilisant les langages HTML, CSS et JavaScript. Ce site est accessible à l'adresse suivante :

<https://soiariis.github.io/Dtres/>

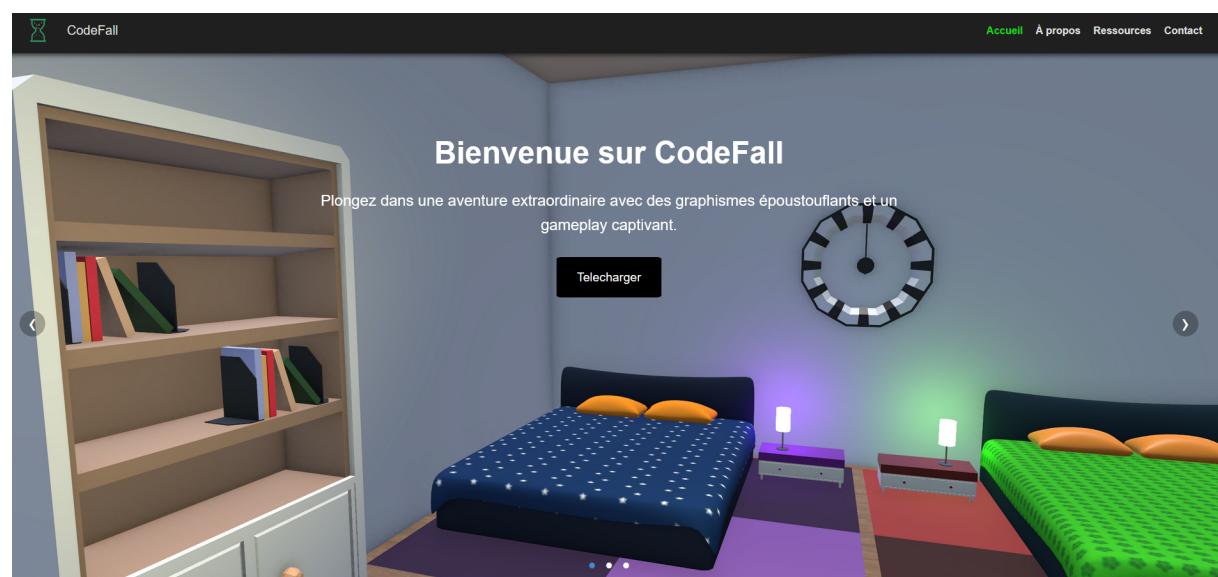
L'architecture du site repose sur une organisation en dossiers :

- un dossier contenant les assets (images et fichiers) ;
- un dossier contenant trois fichiers CSS, chacun dédié à améliorer la lisibilité et l'ergonomie d'une page.

Depuis la dernière soutenance, nous avons modifié l'esthétique générale du site, notamment en changeant la couleur secondaire (bleu remplacé par le vert, en lien avec l'univers de CodeFall).

Le site comporte quatre pages : “Accueil”, “À propos”, “Ressources” et “Contact”.

La page “Accueil” comporte un carrousel avec trois images du jeu, une phrase d'introduction, et un bouton de téléchargement fonctionnel avec un effet “glitch” pour rappeler les bugs du jeu.



La page “À propos” présente l'équipe de développement (Dtres), composée de 4 membres avec les noms et avatars des quatre membres.

Dtres

Notre entreprise est dédiée à la création de jeux immersifs et innovants.



Florian Gagnier
Développeur du networking et du site web



Imane Sarwar
Écriture de l'histoire et IA



Florent Fossati
Développeur en images et ingénieur son



Alexandre Salgueiro
Création des graphismes et options

La page “Ressources” informe les joueurs sur les outils utilisés pour la création du jeu, tels que Unity ou Blender. Elle propose également un accès aux différents documents liés au projet, comme le cahier des charges, les rapports de soutenances ou encore les plans. Le rapport du projet ainsi que le manuel d’installation et d’utilisation ont été ajoutés.



Visual Studio Code



Blender



Git

Cahier des charges fonctionnel

Cahier des charges technique



Github



Photon



Rider

Rapport de projet

Manuel d’installation et d’utilisation



Unity



FL Studio



Overleaf

Première soutenance

Rapport de soutenance 1

Plan de soutenance 1

Deuxième soutenance

Rapport de soutenance 2

Plan de soutenance 2

Troisième soutenance

Plan de soutenance 3

Enfin, une page “Contact” permet aux joueurs de nous signaler d’éventuels bugs, de suggérer de nouvelles fonctionnalités ou simplement de partager leur avis sur le jeu.

Nous contacter

Si vous avez des questions ou des recommandations, n’hésitez pas à nous contacter en utilisant le formulaire ci-dessous.

Votre nom

Votre email



Votre message

Envoyer

Ces pages offrent une vitrine fonctionnelle et claire de notre projet, tout en renforçant l’identité visuelle de CodeFall.

2. Scénario

Le scénario de CodeFall est bien plus qu'un simple fil conducteur narratif. Il constitue en quelque sorte le cœur du jeu, portant à la fois une aventure immersive et un message de fond aux résonances contemporaines. Pensé comme une expérience à la fois ludique et introspective, il invite le joueur à réfléchir sur les choix qu'il fait, les règles qu'il transgresse, et les mondes qu'il altère qu'ils soient réels ou virtuels.

A. Contexte général et mise en situation

L'aventure s'ouvre dans un monde virtuel dont l'esthétique apaise immédiatement. Tout semble harmonieux, maîtrisé, presque utopique. Deux personnages humains émergent dans des chambres. Rien ne leur est encore expliqué, aucun objectif n'est donné. Le joueur partage cette absence de repères, dans un environnement encore silencieux.

C'est alors qu'apparaissent deux petits robots flottants, dotés d'un ton curieux, bienveillant et presque enfantin. Ces guides numériques deviennent leurs compagnons de route : ils communiquent par des messages offrant ainsi indices, rappels de logique, ou juste de simples observations . Dès les premières interactions, on comprend que ces robots ne sont pas de simples outils d'aide : c'est aussi une conscience, une mémoire du monde, peut-être même des témoins de ce qui fut.

Ce monde, qui semblait immuable, commence lentement à montrer des signes de faiblesse. Des anomalies, appelées "bugs", apparaissent petit) petit. Leur nature est déroutante : elles modifient les règles de la physique, contournent les limites spatiales, altèrent la logique même des énigmes. Quand on les utilise avec ingéniosité, ces bugs permettent de progresser... mais à quel prix ?

B. Une narration en trois temps : de la découverte à la désillusion

L'histoire suit une structure tripartite classique. En effet chaque acte reflète non seulement un stade de progression, mais aussi un état psychologique du joueur et du monde qu'il traverse.

Acte I – L'Éveil

Dans cette première phase, tout semble presque parfait. Les décors sont lumineux, la musique douce, les dialogues du robot teintés d'enthousiasme. Le joueur découvre les règles du monde, teste ses limites, apprend à se mouvoir, à résoudre de petites énigmes. Les bugs apparaissent comme des miracles discrets : sauter haut, désactiver un mur invisible. Rien ne semble grave. Au contraire, on a l'impression de jouer avec un système malléable, ludique, vivant.

Mais déjà, le robot glisse ça et là quelques remarques ambiguës : "Les choses ne sont pas censées se passer ainsi", ou encore "Certains chemins ne devraient pas être empruntés". Ce sont les premiers signaux faibles, que peu de joueurs prennent au sérieux.

Acte II – L'Exploitation

À ce stade, les bugs deviennent centraux dans la mécanique du jeu. Ils sont nécessaires à la progression, au point que les joueurs ne peuvent plus avancer sans eux. Le monde, en revanche, commence à se détériorer : textures qui changent, éléments du décor qui se déplacent. L'environnement devient instable, inquiétant.

Le robot, lui aussi, change de ton. Il devient plus hésitant, parfois silencieux. Parfois même inquiet : "Chaque chose modifiée laisse une cicatrice", dit-il. Le jeu commence à "parler" au joueur par des voies détournées, comme si l'univers tentait de se défendre à travers ces anomalies.

Acte III – La Chute

Le dernier acte plonge le joueur dans un monde en ruine. Les lois de la réalité s'affondrent : les objets flottent ou s'effacent, le décor des couloirs devient de plus en plus saccadé. Les bugs, autrefois utiles, produisent maintenant des effets imprévisibles et souvent négatifs : objets mal placés...

Les robots, jadis compagnons de route, sont désormais distants. Parfois accusateur. Il laisse entendre que les dégâts sont irréversibles, et que les joueurs ont franchi une ligne invisible. Les dernières énigmes ne sont plus seulement des défis : ce sont des conséquences. Le joueur ne résout plus des problèmes, il les affronte.

Le jeu s'achève dans une scène figée, glacée, presque silencieuse. Un dernier message s'affiche, lentement, comme arraché à un système mourant : "C'est vous qui détenez le code" ou encore "Réveillez-vous"

C. Dialogues, tonalité et narration indirecte

Les dialogues des robots sont le cœur émotionnel du jeu. Ils suivent une courbe dramatique subtile : de l'enthousiasme à la résignation. Ce changement de ton agit comme un miroir émotionnel du joueur lui-même. Le joueur progresse, mais à mesure que son pouvoir grandit, le monde s'effondre. Et les robots en sont témoin.

D'autres éléments narratifs, plus subtils, viennent enrichir cette immersion :

- Des éléments visuels symboliques (décor aux textures anormales)
- Salles avec des éléments de décor perturbés

Certains moments clés utilisent même des "bugs narratifs", où le jeu semble s'adresser directement au joueur, comme si l'univers avait pris conscience de sa propre disparition.

D. Un message écologique et humain déguisé en gameplay

Le véritable propos de CodeFall ne se cache pas dans ses textes, mais dans sa mécanique même. Chaque bug utilisé est une métaphore : pour aller plus vite, on triche un peu. Pour résoudre un problème, on détourne les règles. Mais chaque raccourci fragilise l'ensemble.

Ce n'est pas un jeu qui cherche à punir, mais à faire réfléchir. Le joueur est libre de continuer. Rien ne l'en empêche. Mais à un moment donné, il comprend que sa progression a un prix. Il n'est pas "le héros" : il est aussi, peut-être, le responsable de l'effondrement.

C'est ce paradoxe qui fait la force du message : on ne cherche pas seulement à faire culpabiliser, mais à provoquer une prise de conscience. Une mauvaise fin volontaire, comme un miroir de notre réalité : celle où nos choix technologiques, économiques ou individuels ont un impact souvent irréversible.

E. Des mécaniques qui racontent autant que les mots

Chaque énigme du jeu a été conçue avec une justification narrative. Aucun bug n'est gratuit. Leur utilisation traduit la disposition du joueur à contourner le monde pour obtenir ce qu'il veut. Et le jeu le lui montre.

Ces situations ne sont pas des effets de style. Elles sont pensées pour rappeler que chaque solution rapide, chaque exploitation, chaque violation du système, laisse une trace.

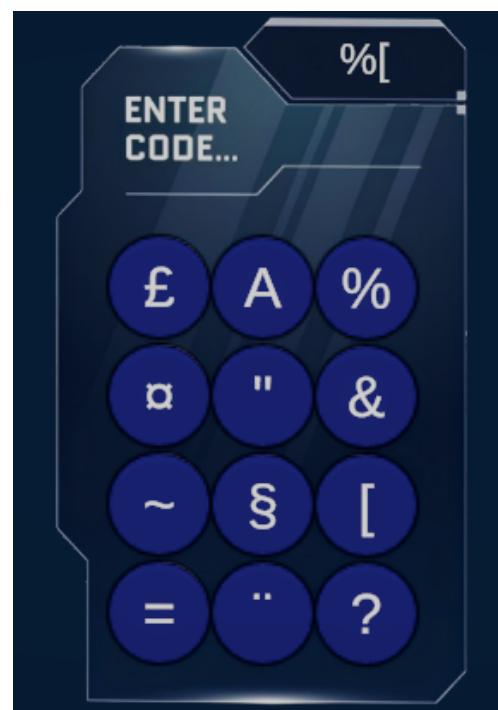
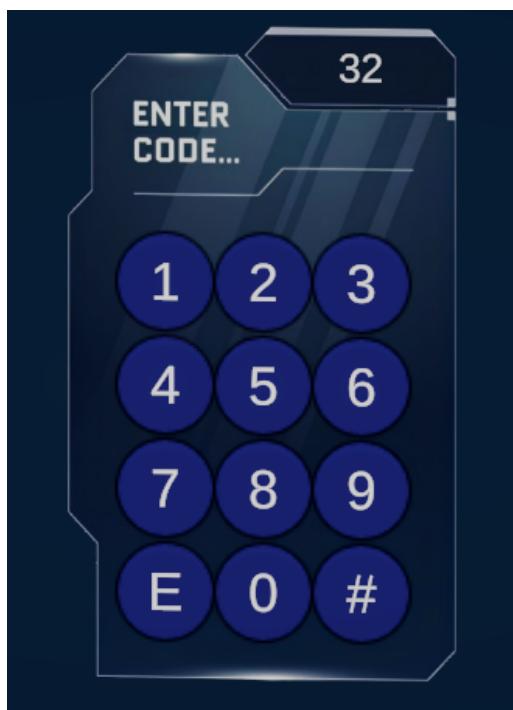
3. Mécaniques de jeu/Gameplay

A. Portes

Deux types de portes ont été intégrés dans le jeu :

- Portes simples : elles s'ouvrent automatiquement dès qu'un joueur entre dans leur collider. Le système ne nécessite pas de code. Ce type de porte a été rapide à implémenter.
- Portes à code : elles comportent une interface de digicode. Lorsqu'un joueur appuie sur la touche "E" à proximité, il peut entrer un code de 4 caractères. Si le code est correct, la porte s'ouvre avec une animation ;sinon, les caractères sont réinitialisés. Le bouton "#" permet d'effacer la saisie.

Une variante spéciale a été créée pour la bibliothèque, utilisant des caractères représentant des valeurs (ex. : A = 10, = +). Le joueur doit alors entrer une équation donnant un résultat précis. Le système vérifie dynamiquement si l'équation est correcte avant d'ouvrir la porte.





Donc la porte dans ce cas va faire le calcule des valeurs donné pour voir si le résultat est celui demandé pour la porte. Si cela est le cas, la porte va s'ouvrir.

Une texture d'écran est placée aux endroits auxquels les joueurs peuvent mettre des codes. La plupart du temps ces écrans sont mis directement sur une porte cependant parfois le digicode peut loin de la porte.

Pour faire les portes nous avons dans un premier temps suivis des tutoriels. Malgré cela, le développement de cette mécanique a d'abord été laborieux. De plus cette première version à été imparfaite. Tout au long du développement du jeu, le code de la porte a évolué et est devenu de plus en plus efficace car notre compréhension a progressé au fil du temps. Nous avons ainsi pu optimiser le code des portes pour le rendre plus robuste et efficace.

B. Livres

Des livres interactifs sont disséminés dans le jeu. Ils permettent de transmettre des indices ou des éléments de scénario.



Techniquement, chaque livre possède un canvas attaché qui s'active au clic. Ce système utilise un “raycast” lancé depuis la caméra du joueur. Le script est conçu pour fonctionner uniquement en local afin d'éviter d'interrompre l'expérience de l'autre joueur.



La création des livres a été relativement simple grâce aux similitudes avec les portes. Par la suite, le seul aspect du livre qui a dû être retravaillé est l'apparence et les textures.

C. Dialogues

Pour donner vie aux robots, nous avons mis en place un système de communication entre le robot et le joueur, via une interface dédiée et une intervention vocale faite de “bip” et “bop”, caractéristiques des robots et représentant leur nom respectif. Cela leur donne une personnalité, ce qui permet au joueur de mieux s’immerger dans l’univers du jeu. Ces sons transforment les robots en entités actives, perçues comme conscientes ou autonomes, ce qui enrichit l’expérience narrative.

Certains dialogues peuvent fournir au joueur des indications sur l’environnement, des alertes ou des réactions à ses actions. Des répliques bien placées (humoristiques, dramatiques, logiques, etc.) donnent aux robots une forme d’humanité, favorisant l’attachement du joueur. Cela peut renforcer l’impact émotionnel de certaines scènes du jeu. En déclenchant ces dialogues à des moments clés, on évite les temps morts et on maintient une dynamique narrative vivante. Cela contribue à casser la monotonie et à maintenir l’attention du joueur.

L’ajout de dialogues contextuels pour les robots permet d’enrichir l’univers du jeu, de favoriser l’immersion, d’informer le joueur et de créer un lien narratif et émotionnel avec des entités non humaines. Il s’agit d’un outil narratif et ludique puissant, qui améliore l’expérience globale de jeu.



Cette interface affiche l'avatar du robot (différent selon le joueur), donnant ainsi l'impression que celui-ci s'adresse directement au joueur. Certains événements déclenchent automatiquement des dialogues. Le joueur peut utiliser un clic droit pour accélérer l'affichage du texte ou passer au message suivant.

Le joueur peut appuyer sur "A" pour demander un indice au robot, par la suite un dialogue apparaîtra avec l'indice de la salle si le joueur est dans une salle où il faut résoudre une énigme.

D. Caisses

Nous pensions que résoudre le problème de la caméra permettrait de faire fonctionner correctement la mécanique de port de caisse. Pour rappel, lors de la dernière soutenance, nous pouvions déplacer une caisse, mais cela ne fonctionnait pas en multijoueur. Nous avions supposé que synchroniser les caméras réglerait le problème, or ce n'était pas le cas. C'est pourquoi nous avons dû entièrement retravailler le système de la caisse.

Désormais, les joueurs peuvent prendre n'importe quel objet du jeu, à condition qu'il possède le tag : "CanBeMove". Lorsqu'un joueur saisit une caisse, celle-ci se positionne automatiquement devant lui.



Nous avons décidé de garder le box collider de la caisse activé lorsque le joueur la porte. Nous avons fait ce choix afin de pouvoir créer un “bug de collision” intentionnel.

En effet, comme mentionné tout au long du projet, la spécificité de CodeFall est que les joueurs doivent parfois utiliser des “bugs” pour progresser. Dans le cas de la caisse, nous l'avons codée spécialement pour permettre trois types de “bugs” :

- Le premier est un bug de collision simple : si un joueur se tient sur la caisse pendant que l'autre l'attrape, la hitbox du joueur et celle de la caisse vont se superposer, ce qui propulsera le joueur dans les airs.
- Le deuxième bug permet au joueur de porter la caisse à travers les murs. Si le mur est suffisamment fin, la caisse peut dépasser de l'autre côté, ce qui permet de la faire passer à travers.
- Le troisième bug est révélé au joueur à la fin du jeu. Il consiste à pousser un joueur avec une caisse contre un mur afin de le faire passer à travers. Ce bug ne fonctionne que si le joueur est bloqué dans un coin, car sinon il sera simplement poussé là où la caisse ne se trouve pas.

Ces exploits sont intentionnels, car ils font partie des mécaniques de jeu intégrées et expliquées au joueur. Si nous avions voulu les empêcher, nous aurions désactivé la hitbox de la caisse lorsqu'elle est portée, et nous aurions fait en sorte que les murs et les joueurs repoussent celle-ci.

Cependant, nous savons que ces exploits peuvent entraîner des complications. C'est pourquoi nous avons prévu une sécurité : si une caisse est déplacée à travers un mur et tombe dans le vide, elle est automatiquement téléportée à côté du dernier joueur qui l'a portée.

Pour cela, nous avons ajouté un script au joueur qui enregistre la caisse et vérifie sa position dans l'espace. Si la position est trop basse, le script la réinitialise. Afin d'éviter que les joueurs puissent sortir de la carte, nous avons également fait en sorte qu'ils ne puissent pas utiliser le premier bug en extérieur.

E. Indices

Tout au long du jeu, les joueurs peuvent se retrouver perdus ou bloqués. C'est pourquoi nous avons ajouté un système d'aide, activable à la demande. Il suffit aux joueurs d'appuyer sur la touche "A" de leur clavier pour obtenir un indice de la part de leur robot. En fonction de la salle, l'indice peut être plus ou moins subtil.



Pour implémenter cela, nous avons utilisé des box collider, quand le joueur est dans l'un d'entre eux, le dialogue du robot qui s'affiche en demandant un indice, change.

Cette mécanique a été surprenante facile à implémenter, effectivement nous l'avons ajouté vers la fin de la création du jeu. Grâce à cela nous savions bien comment unity fonctionnait et comment implémenter cette mécanique de manière efficace.

4. Enigme

A. Spawn

Le spawn est la première salle que les joueurs rencontrent. Elle contient l'éénigme la plus simple du jeu : un livre dans lequel se cachent des chiffres qui, une fois mis dans l'ordre, composent le code de la première porte. Cette éénigme très simple a pour but de montrer au joueur comment fonctionnent la porte et le livre. En effet, cette salle fait office de tutoriel.

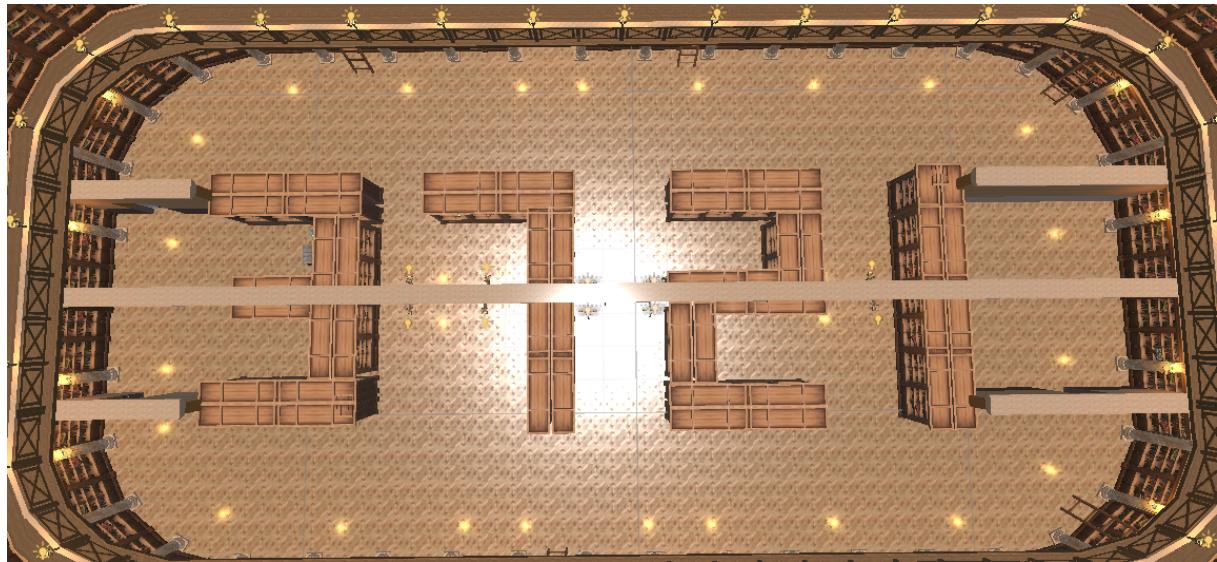


B. Bibliothèque



La bibliothèque commence dans une salle contenant trois portes à code, chacune avec une valeur affichée au-dessus. Pour ouvrir ces portes, les joueurs doivent lire les différents livres cachés dans la pièce. Ces livres permettent de décrypter les digicodes : les joueurs vont comprendre que ces digicodes fonctionnent comme des calculatrices prenant deux valeurs et un signe (division, addition, soustraction, etc.). Ils devront alors calculer les valeurs affichées au-dessus des portes en identifiant correctement à quoi correspondent chaque signe.

Dans la deuxième partie de la bibliothèque, les joueurs doivent réaliser leur premier “bug”. Pour obtenir le dernier code, l’un des joueurs devra prendre de la hauteur.



Cependant, cela n'est pas possible par des moyens conventionnels. C'est pourquoi, guidés par un indice, les joueurs vont exécuter un “bug” de collision avec une caisse pour s'elever dans les airs.

Les caisses ont été codées spécialement pour permettre la réalisation de ce “bug”, car le jeu prévoit que les joueurs doivent “casser” les règles de ce monde.

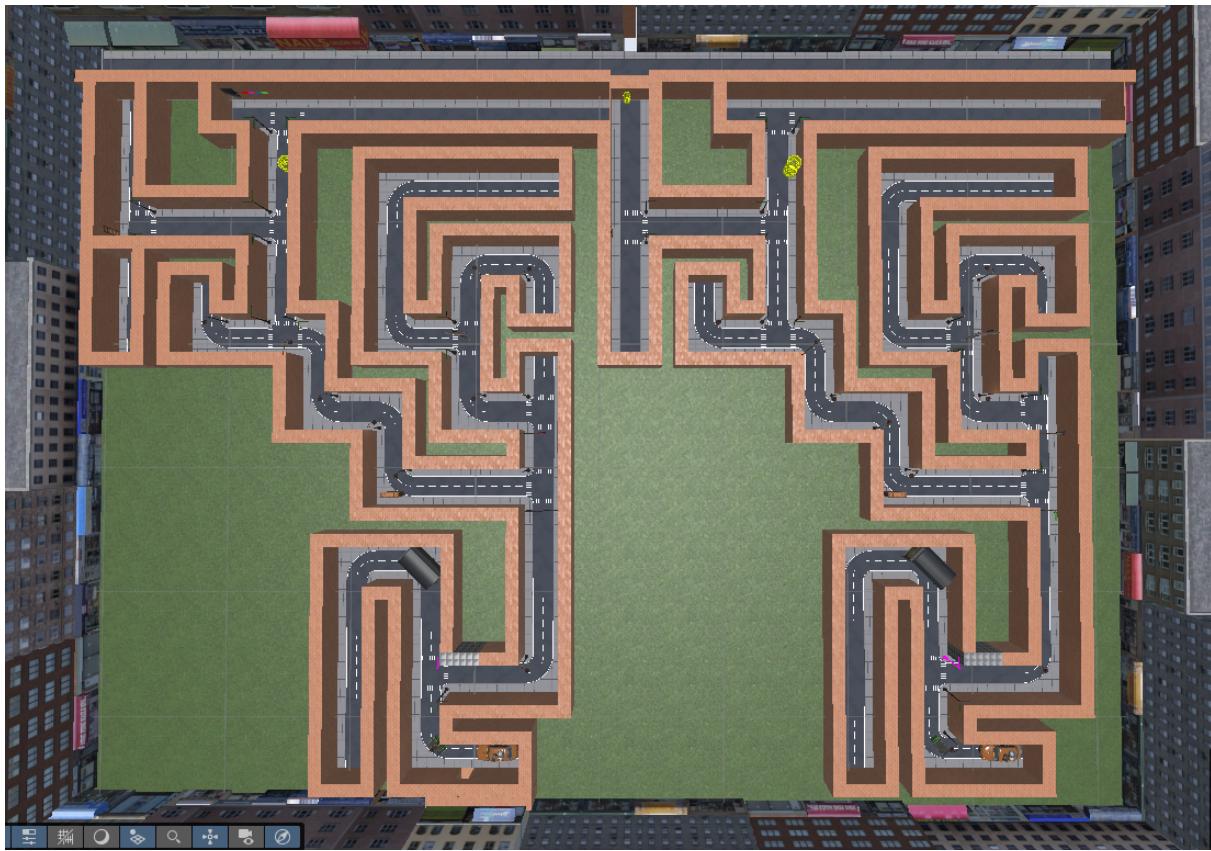
Grâce au code obtenu, les joueurs pourront ouvrir la porte et accéder à la dernière salle.

C. Labyrinthe

Le labyrinthe est la plus grande salle du jeu. Pour la traverser, les joueurs devront communiquer. En effet, ils seront séparés : l'un des joueurs sera téléporté dans un labyrinthe. En réalité, chacun des deux joueurs se retrouvera dans un labyrinthe identique, mais distinct.

En explorant leur labyrinthe, les joueurs découvriront un mur qui a une texture, mais pas de box collider. Ils pourront donc passer à travers ce “faux mur” et y trouver une caisse ainsi qu'un plan de la salle. Pour aider à repérer ce mur, nous avons placé des indices visuels.

En examinant le plan, les joueurs vont se rendre compte qu'un mur très fin les sépare en haut de la salle. En combinant cela avec des informations trouvées dans un livre caché à gauche de la salle, ils vont pouvoir réaliser le deuxième “bug” : faire passer une caisse à travers les murs lorsqu'ils sont fins.



Après cela, l'un des deux joueurs pourra sortir. Il devra alors résoudre une énigme rapide en communiquant avec l'autre joueur, afin de lui ouvrir une porte de sortie et ainsi accéder à la dernière salle d'énigme du jeu.



Le but de cette salle est d'introduire le principe de coopération dans le jeu. Là où les deux premières salles pouvaient être résolues presque seul, celle-ci nécessite impérativement la communication.

D. Les salles de classes

La dernière salle d'énigme est divisée en quatre salles de classe.

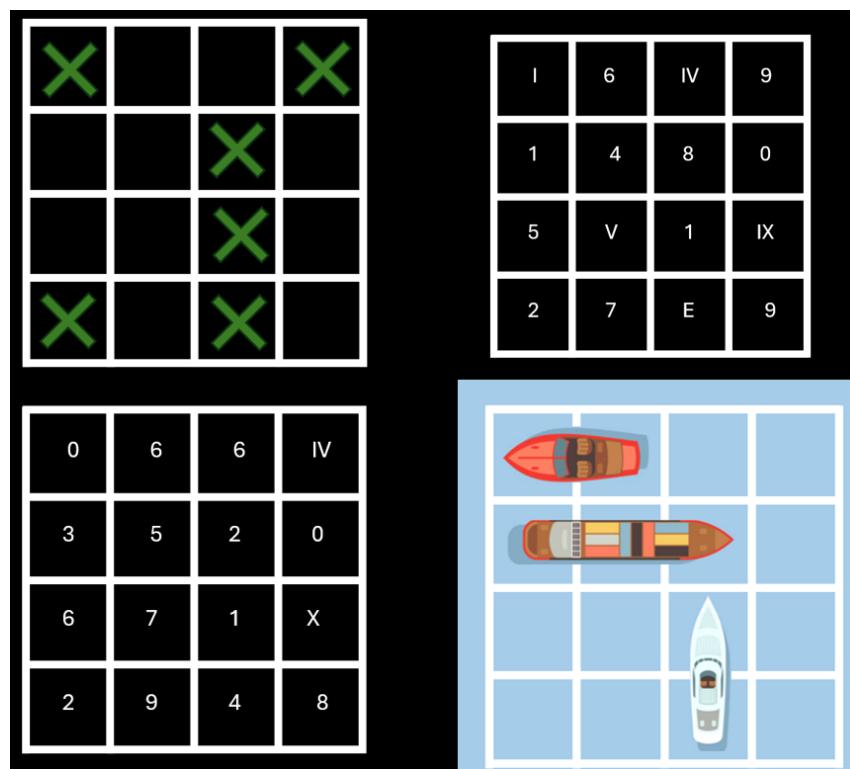


Il faut savoir qu'il y a, au total, quatre portes dans ce niveau : les trois portes rouges sont des portes à code, et la porte orange est la porte de fin, qui ne peut s'ouvrir que depuis l'autre côté. Le joueur "client" est téléporté dans la salle de classe en bas à gauche. Il se retrouve donc séparé de l'autre joueur à cause d'une porte à code. Pour progresser, la seule chose que les joueurs peuvent faire est de trouver le code caché sur les tableaux, puis de le saisir sur le digicode situé à gauche du niveau.

Le joueur client va être téléporté dans la salle de classe en bas à gauche. Il va donc se retrouver séparé de l'autre joueur à cause d'une porte à code.

Cependant, une fois le code entré, ce n'est pas la porte de gauche qui s'ouvre, mais celle de droite. Cela s'explique, dans le scénario, par la dégradation progressive du monde.

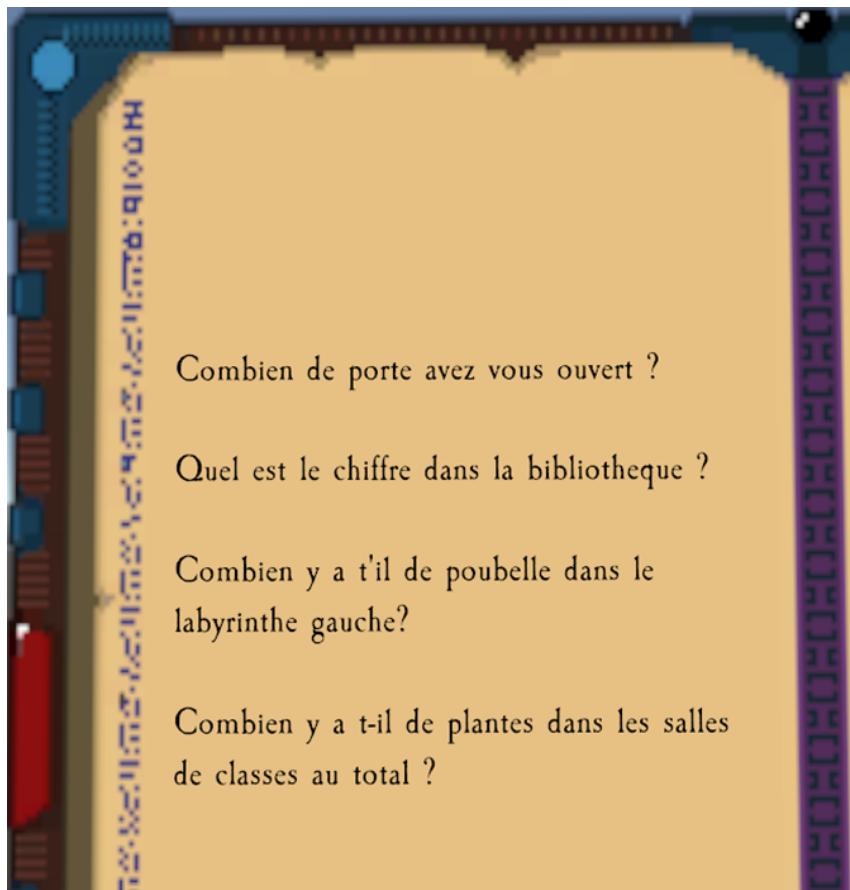
À ce stade, les joueurs ont accès à une caisse, un digicode et quatre tableaux.



Avec ces tableaux, les joueurs vont déduire qu'ils représentent une partie de bataille navale. Ils vont ainsi trouver le code "181E". En entrant ce code, la porte en haut à gauche va s'ouvrir, révélant la dernière salle. Désormais, il ne reste plus qu'une seule porte à code qui sépare les deux joueurs.

Dans cette dernière salle, un tableau explique comment réaliser le troisième bug de la caisse, et un livre donne des indices pour trouver le dernier code du jeu.

Pour trouver ce dernier code du jeu, les joueurs vont devoir comprendre ceci :



En effet, les joueurs vont devoir revenir en arrière pour compter les objets un à un à travers le jeu. Une fois cela fait, ils pourront se retrouver et réaliser le bug indiqué sur le tableau, afin d'accéder à la fin du jeu.

La réalisation des salles a été longue et fastidieuse. Nous avons dû refaire entièrement le labyrinthe et la bibliothèque. Il a fallu fournir un effort important de réflexion pour trouver des idées neuves et des enchaînements logiques. Le plus difficile a été d'ajuster la difficulté des énigmes. En effet, l'étape la plus délicate lorsqu'on conçoit une énigme est d'en évaluer la difficulté.

Pour surmonter ce problème, nous avons demandé l'avis de nos amis et de nos proches sur nos énigmes, ainsi que leurs suggestions. Finalement, nous sommes fiers du résultat obtenu.

5. Multijoueur

Afin de comprendre le fonctionnement du multijoueur en mode peer-to-peer (P2P), j'ai visionné plusieurs vidéos explicatives, notamment sur YouTube. Le modèle P2P repose sur le principe selon lequel chaque joueur agit à la fois comme client (pour envoyer des données) et comme serveur (pour en recevoir).

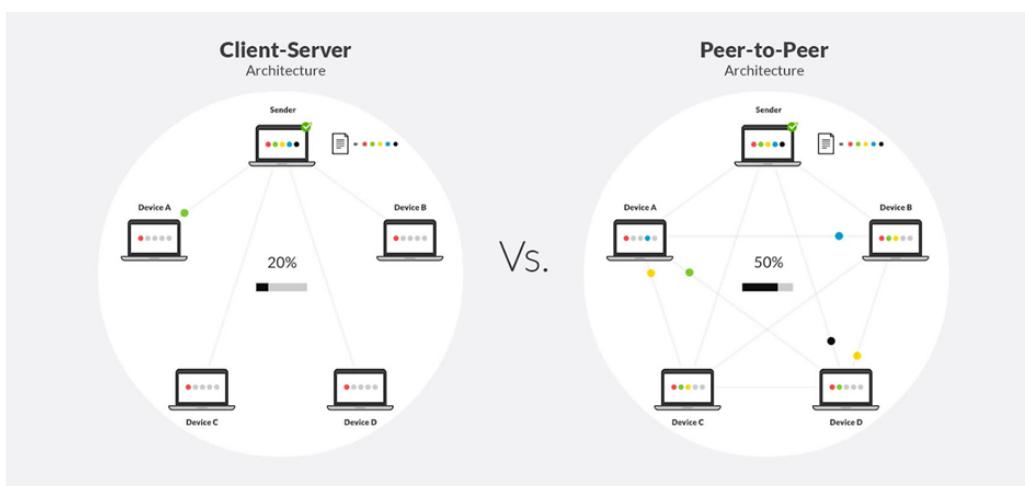
Parmi les différentes options disponibles pour gérer le réseau, nous avons choisi d'utiliser l'outil Photon Fusion, réputé pour sa popularité et ses nombreuses fonctionnalités. Fusion propose un système dans lequel un joueur peut agir comme hôte, tandis que les autres se connectent en tant que clients. Ce modèle P2P, associé à une architecture client-serveur décentralisée, nous a semblé particulièrement adapté au jeu multijoueur d'éénigmes que nous développons.

Avantages du modèle P2P pour notre projet :

- Coûts réduits : pas besoin d'héberger des serveurs dédiés.
- Simplicité de création de sessions : les joueurs peuvent créer des parties sans dépendre de serveurs officiels.
- Expérience sans latence significative : l'hôte bénéficie souvent d'une meilleure fluidité de jeu.

Inconvénients acceptables pour notre type de jeu :

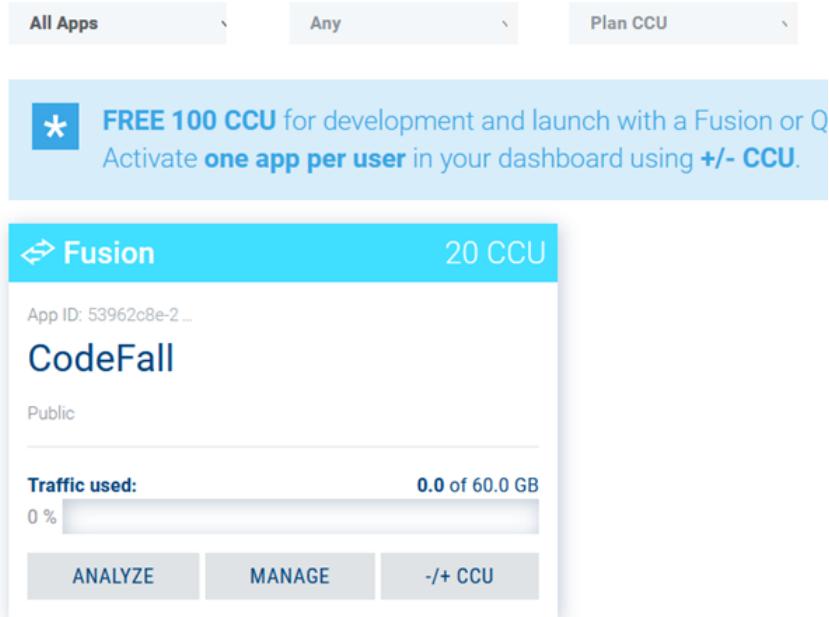
- Échelle limitée : le jeu est prévu pour deux joueurs seulement, donc ce n'est pas une contrainte.
- Avantage en termes de latence pour l'hôte : dans un jeu coopératif, cela n'est pas problématique.
- Facilitation de la triche : ce risque est négligeable dans un jeu d'éénigmes coopératif.



Après avoir installé Photon Fusion, j'ai consulté des vidéos explicatives et exploré la documentation officielle pour en comprendre le fonctionnement, notamment la configuration du projet à l'aide d'une AppID.

<https://doc.photonengine.com/fusion/current/fusion-intro>

J'ai également étudié de nombreux projets existants développés avec Fusion, ce qui m'a permis de comprendre les concepts clés et d'ajouter des fonctionnalités au jeu. Par exemple, j'ai pu mettre en place un système permettant de faire apparaître un personnage après la création du serveur ou de la salle.



L'un des aspects les plus complexes et formateurs de ce projet a été le développement de la gestion des déplacements des personnages en multijoueur, combinée à une vue à la première personne (First Person View). Ce travail m'a permis de plonger au cœur des problématiques réseau, de la synchronisation d'état et de l'interaction joueur-caméra dans un environnement en temps réel.

Pour permettre une expérience fluide et immersive, j'ai dû implémenter un système dans lequel les joueurs se déplacent en fonction de la direction dans laquelle leur propre caméra regarde. Cela peut sembler trivial dans un jeu solo, mais en multijoueur, cela requiert une synchronisation fine entre les données locales (input, rotation de la caméra) et leur diffusion correcte à travers le réseau.

A. Interface

Pour structurer l'accès au jeu, j'ai développé une interface conviviale sous forme de menu principal. Elle permet aux joueurs de :

- saisir un pseudonyme et le nom d'une salle de jeu ;
- créer une salle en tant qu'hôte (serveur) ;
- être automatiquement redirigés vers une scène de lobby en attente d'un second joueur.

Dans le lobby, les fonctionnalités suivantes ont été implémentées :

- affichage dynamique des pseudos et du nom de la salle ;
- activation conditionnelle du bouton “Start” uniquement lorsque les deux joueurs sont connectés ;
- redirection automatique vers la scène de jeu une fois le bouton “Start” pressé.

Ce système assure une transition fluide et intuitive entre la phase de connexion et le début de la partie.

B. Déconnexion

J'ai également intégré un système robuste de gestion de la déconnexion. Si l'un des deux joueurs quitte la partie (volontairement ou non), la session est automatiquement interrompue pour les deux, et les utilisateurs sont redirigés vers le menu principal. Cela évite tout état incohérent du jeu et renforce l'aspect collaboratif de l'expérience.

C. Caméras

L'un des défis techniques majeurs a concerné la gestion des caméras. Initialement, une seule caméra — celle créée en dernier — restait active, quelle que soit la position du joueur. Cela provoquait une inversion des perspectives :

- L'hôte voyait à travers la caméra du second joueur.
- Les mouvements du second joueur suivaient l'orientation de la caméra de l'hôte, et non la sienne.

Ce bug s'est révélé particulièrement difficile à diagnostiquer. Pendant longtemps, nous ne parvenions pas à en identifier la source. Ce n'est qu'après de longues recherches sur des forums spécialisés, des tutoriels vidéo et la lecture de documentation technique que nous avons compris l'origine du problème : une mauvaise gestion de la synchronisation des rotations de caméra entre les instances du réseau.

Plus précisément, il s'agissait d'un mauvais renvoi des données de rotation de la caméra dans le système de synchronisation. Cela provoquait une discordance entre la direction réelle d'un joueur et celle perçue par son propre client ou par l'hôte. Une fois ce problème isolé, sa correction a permis de restaurer un comportement normal, où chaque joueur contrôle sa propre caméra et se déplace en fonction de la direction qu'il regarde.

La résolution de ce bug a été extrêmement gratifiante. Elle m'a permis de renforcer mes compétences non seulement en programmation réseau et en débogage, mais aussi en persévérance et en méthodologie de recherche. Ce type de problème illustre à quel point une fonctionnalité apparemment simple peut cacher une grande complexité, en particulier dans un environnement multijoueur en temps réel.

D. Lobby

Dans le cadre de la conception du lobby multijoueur et des éléments interactifs synchronisés, plusieurs défis techniques ont émergé, notamment autour de la cohérence des données partagées entre les joueurs. Ces problématiques, bien qu'en apparence mineures, sont cruciales pour garantir une expérience utilisateur fluide et immersive.

Un des premiers problèmes identifiés concernait l'affichage des pseudonymes des joueurs dans le lobby. Initialement, chaque joueur ne voyait que son propre pseudo : l'hôte ne voyait que le sien, tout comme le client. Cette limitation compromettait la clarté de l'interface, rendant difficile la confirmation de la présence effective des deux joueurs dans la salle.

Ce problème provenait du fait que chaque joueur exécutait la méthode **Update-PlayerNameUI()** avec uniquement les informations disponibles localement. Ainsi, chacun envoyait son propre pseudo, sans connaître celui de l'autre.

Pour résoudre ce problème, nous avons modifié la logique afin que seul l'hôte — qui possède une vue complète sur tous les joueurs connectés — exécute cette méthode. Il envoie ensuite les pseudonymes synchronisés à chaque client via un Remote Procedure Call (RPC) ciblé. De cette manière, chaque joueur reçoit la liste complète et correcte des participants dans le lobby.

Une autre amélioration importante dans le lobby concernait la synchronisation visuelle du point de départ sélectionné par l'hôte. Nous avions conçu une interface où l'hôte

pouvait choisir un emplacement de spawn, mais initialement, ce changement n'était visible que de son côté. Le client ne voyait donc pas les modifications apportées.

Nous avons mis en place une synchronisation en temps réel de cette sélection. Chaque changement effectué par l'hôte est désormais envoyé via RPC et mis à jour sur l'interface du client. Ainsi, les deux joueurs voient exactement le même environnement et partagent une compréhension commune du point de départ choisi avant le lancement de la partie.



E. Porte

L'un des éléments clés du gameplay repose sur des interactions synchronisées, comme l'ouverture d'une porte après la résolution d'une énigme.

Lorsque l'un des joueurs entre un code correct, la porte doit s'ouvrir simultanément pour les deux joueurs. Or, sans gestion réseau appropriée, l'action n'était visible que du côté de l'initiateur, provoquant une désynchronisation de l'état du jeu entre client et hôte.

Nous avons ajouté un NetworkObject au prefab de la porte afin qu'elle soit reconnue par le système réseau comme un objet synchronisable. Cela nous a permis d'utiliser des appels RPC pour gérer son état.

Lorsque l'un des deux joueurs entre le bon code, le script vérifie s'il possède l'autorité sur l'état (HasStateAuthority).

- Si c'est le cas, alors le joueur est l'hôte, et il appelle directement une RPC_OpenDoor, qui déclenche l'animation d'ouverture de la porte et désactive le collider pour permettre le passage.

- Si c'est un client, il appelle **RPC_RequestOpenDoor**, qui vérifie la demande, puis appelle à son tour **RPC_OpenDoor**.

Ainsi, quel que soit le joueur qui entre le bon code, l'ouverture de la porte est toujours initiée par l'hôte, garantissant une synchronisation parfaite pour tous les participants.

F. Caisse

L'un des objets interactifs du jeu est une caisse pouvant être saisie, déplacée et relâchée par les joueurs. En multijoueur, la gestion de cet objet s'est révélée particulièrement complexe.

- Lorsqu'un joueur saisissait la caisse, l'action n'était pas visible pour l'autre joueur.
- La position de la caisse pouvait diverger entre l'hôte et le client.
- Les actions de saisie et de relâchement n'étaient pas toujours synchronisées, générant des comportements incohérents.

Nous avons commencé par ajouter un NetworkTransform au prefab de la caisse, afin de synchroniser sa position et sa rotation sur tous les clients. Un système basé sur des RPC a également été mis en place pour transmettre les actions de saisie et de relâchement.

Malgré ces ajouts, les actions restaient partiellement désynchronisées. Des effets de latence et des conflits d'autorité entre les instances client et hôte entraînaient encore des erreurs visuelles et logiques.

Après de nombreuses recherches, nous avons conçu un script réseau dédié à la gestion de la caisse, dont le fonctionnement est le suivant :

Saisie de la caisse :

- Lorsqu'un joueur tente de saisir la caisse, un RPC est envoyé au serveur pour demander l'autorisation.

Si la demande est validée, le serveur émet un second RPC à tous les clients pour notifier l'action.

La caisse devient alors "possédée" par le joueur demandeur, et sa position est gérée en fonction de ses déplacements.

Relâchement de la caisse :

- Un RPC est également déclenché et diffusé à tous les clients.
- L'objet est alors rendu à nouveau libre et contrôlé par la physique réseau (via NetworkTransform).

Ce système garantit une synchronisation parfaite des états et de la position de la caisse pour l'ensemble des joueurs.

G. Robots

Pour synchroniser le déplacement et la position des robots pour les deux joueurs, nous avons simplement utilisé un NetworkTransform et remplacé la méthode **Update()** par **FixedUpdateNetwork()**.

H. Animations

Il a également fallu faire en sorte que chaque joueur voie les animations de l'autre.

Au début, les joueurs voyaient l'animation de marche de l'autre uniquement lorsque celui-ci bougeait, mais cela n'était pas toujours cohérent. Nous avons donc ajouté un NetworkBehaviour au script d'animation des personnages et remplacé **Update()** par **FixedUpdateNetwork()**.

Cependant, bien que le client voie correctement sa propre animation et celle de l'hôte, ce dernier ne voyait que sa propre animation. Nous avons ensuite compris qu'il fallait utiliser un NetworkMecanimAnimator en plus d'un Animator, et l'intégrer correctement dans le script pour une synchronisation complète.

I. Conclusion

Le développement du mode multijoueur a représenté l'un des volets les plus techniques et enrichissants du projet. Il a nécessité une compréhension approfondie des principes de synchronisation réseau, de gestion des rôles client/hôte, et de transmission fiable des données entre joueurs.

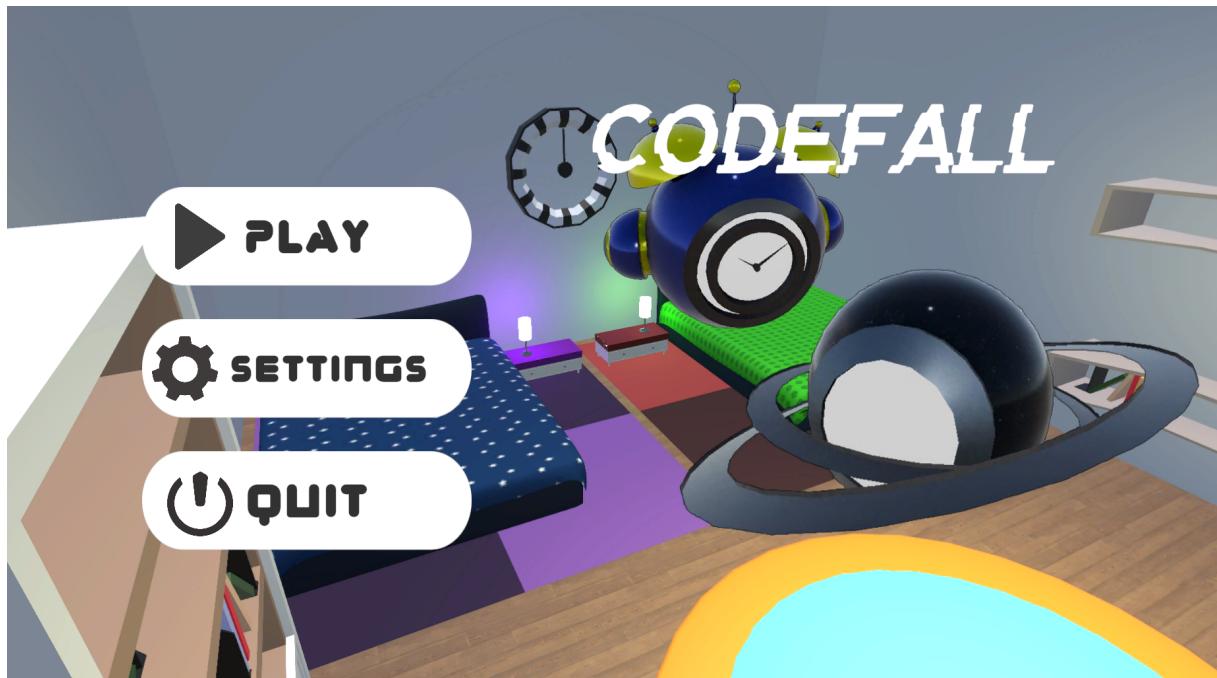
Qu'il s'agisse de la synchronisation des caméras, des objets interactifs comme la caisse ou la porte, ou encore des animations des personnages, chaque fonctionnalité a posé des défis spécifiques.

Ces problématiques nous ont permis de progresser significativement dans l'utilisation des outils réseau, notamment les RPC, NetworkTransform, NetworkMecanimAnimator et la gestion de l'autorité. Les solutions apportées ont permis d'offrir une expérience multijoueur fluide, cohérente et immersive, et nous ont donné une base solide pour de futurs développements en réseau.

6. Interface Utilisateur

A. Menus

Nous avons implémenté un menu au démarrage du jeu, permettant au joueur de choisir entre jouer, accéder aux options, ou quitter. L'interface et le style visuel ont été largement modifiés afin d'offrir une expérience à la fois ergonomique et esthétique.



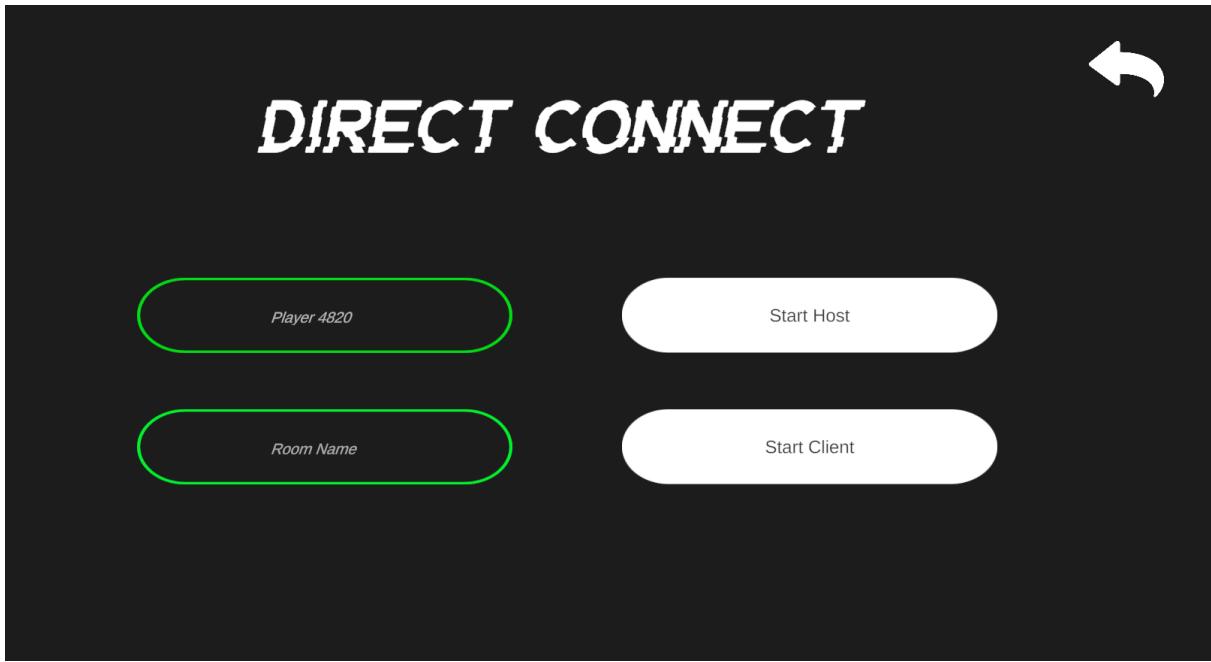
Le menu fonctionne correctement et reste accessible pendant le jeu. Le joueur peut l'ouvrir à tout moment en appuyant sur la touche Échap. Une fois le menu affiché, il peut accéder aux paramètres, quitter la partie ou simplement reprendre le jeu.

Lorsqu'un joueur met le jeu en pause, ses déplacements ainsi que le contrôle de la caméra sont désactivés. Pour cela, nous avons ajouté une vérification dans la méthode responsable du déplacement du joueur et dans celle gérant le mouvement de la caméra, afin de détecter si le menu pause est actif et de bloquer toute interaction liée au mouvement ou à la caméra.

Lorsque le joueur est en train de jouer, le curseur de la souris est masqué et verrouillé au centre de l'écran. Il est visuellement remplacé par un simple point servant de réticule, afin de faciliter l'interaction et la visée sans nuire à l'immersion. Ce comportement permet une expérience plus fluide et intuitive, notamment dans un environnement à la première personne.

Cependant, lorsque le joueur met le jeu en pause ou ouvre une porte, le curseur de la souris réapparaît à la place du point central. Cela permet au joueur de naviguer dans les menus, d'interagir avec l'interface ou de cliquer sur le digicode de manière intuitive.

En plus du menu principal, un menu dédié à la création ou à la recherche de salle est également présent, offrant une interface à la fois intuitive et agréable à utiliser.



Ce menu permet au joueur de créer une nouvelle session de jeu ou de rejoindre une salle existante. Une fois connecté, le joueur est redirigé vers une scène de lobby, où les deux participants attendent avant le lancement de la partie.

Dans ce lobby, les noms des deux joueurs sont affichés afin de confirmer leur présence. Une liste déroulante permet à l'hôte de sélectionner la salle de départ, mais cette option est désactivée côté client pour éviter toute modification non autorisée. Enfin, un bouton Start est disponible uniquement pour l'hôte, et reste désactivé pour le client.



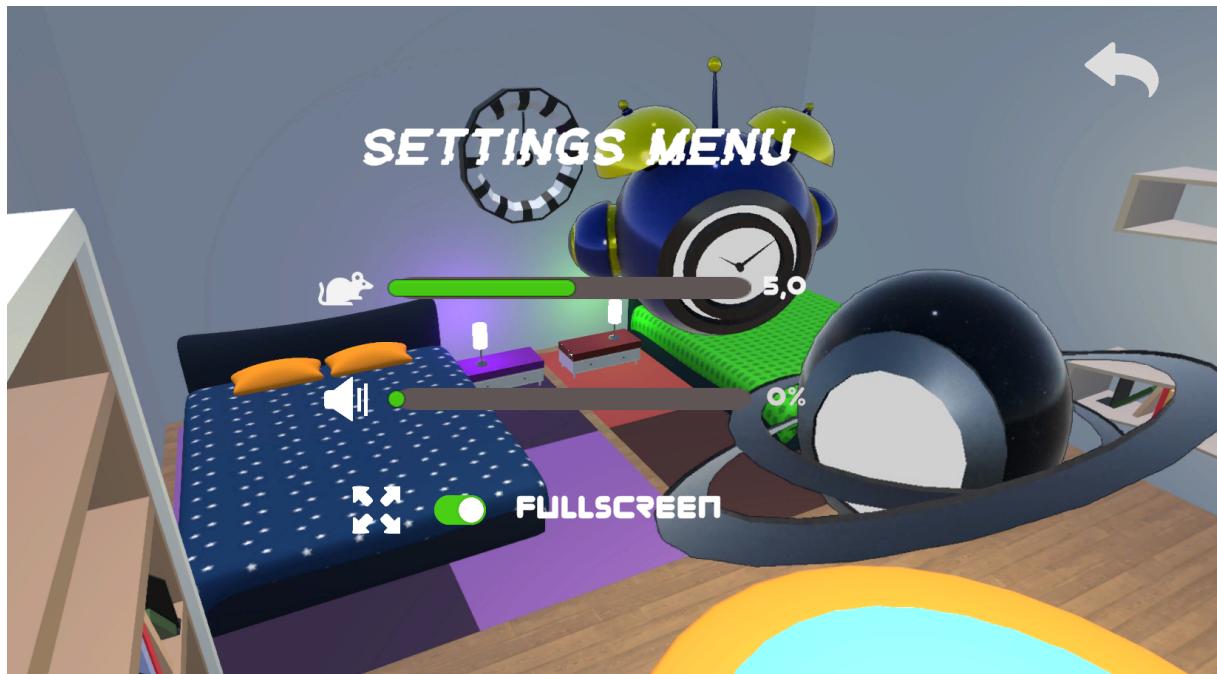
Dans le cadre de notre projet de jeu vidéo, le choix des typographies a été soigneusement réfléchi afin de renforcer l'identité visuelle du jeu et de soutenir son ambiance générale. Trois polices principales ont été sélectionnées : Zorque, GlitchGoblin et Neuropol. Chacune d'entre elles remplit un rôle précis et contribue à l'atmosphère globale de l'expérience de jeu.

- GlitchGoblin, utilisée pour les titres et les noms des joueurs, est une police au style pixellisé et glitché, qui évoque l'esthétique de CodeFall et ses bugs visuels. Ce choix apporte une touche originale et immersive, en cohérence avec l'univers du jeu, tout en attirant l'attention sur les éléments clés comme les identifiants des joueurs ou les intitulés majeurs.
- Zorque est une typographie futuriste et décorative, choisie pour les textes présents dans les boutons ainsi que pour le nom des salles. Son aspect anguleux et audacieux renforce la dimension dynamique et technologique du jeu. Elle permet de rendre ces éléments interactifs plus visibles et esthétiquement marqués, tout en conservant une bonne lisibilité.
- Enfin, Neuropol est utilisée pour l'ensemble des autres textes. Cette police moderne, géométrique et épurée garantit une excellente lisibilité tout en s'intégrant parfaitement dans l'ambiance futuriste du jeu. Elle joue un rôle de support visuel, apportant clarté et cohérence à l'interface globale.

B. Option

Les options permettent actuellement de modifier la sensibilité de la souris, le mode plein écran ainsi que le volume général du jeu. Ce menu a fait l'objet d'une refonte visuelle importante, afin d'offrir une interface plus claire, moderne et agréable à utiliser.

Chaque paramètre est facilement accessible et ajustable en temps réel, ce qui permet au joueur de personnaliser son expérience de jeu selon ses préférences.



C. Crédit

Nous avons ajouté, à la fin du jeu, une scène de crédits afin de présenter aux joueurs les personnes ayant participé au projet, tout en les avertissant de la fin de l'aventure. Cette scène permet de clore le jeu sur une conclusion à la fois surprenante et dramatique, tout en remerciant le joueur pour son engagement dans l'expérience.



7. Intelligences artificielles

Nous avons programmé une intelligence artificielle de type pathfinder qui contrôle les deux robots du jeu. Celle-ci utilise une formule pour calculer la distance qui la sépare du joueur auquel elle est assignée. À partir de cette distance, elle trace un vecteur en trois dimensions pour déplacer le robot vers le joueur.

Cet algorithme de déplacement a été complexe à coder. En effet, il nous a fallu du temps pour comprendre le fonctionnement du déplacement d'un objet dans un espace tridimensionnel. Cependant, le script contrôlant le robot a été ajouté très tôt dans le développement du jeu, ce qui nous a grandement soulagés. Le codage d'une intelligence artificielle nous semblait au départ insurmontable, tant la tâche paraissait complexe.

D'ailleurs, la première version du pathfinder faisait trembler le robot. En effet, les calculs de notre algorithme étaient trop précis, ce qui provoquait des mises à jour constantes de la position du robot. Pour résoudre ce problème, nous avons ajouté une condition : lorsque le robot est suffisamment proche du joueur, il devient immobile. Cette amélioration a grandement amélioré le rendu visuel du robot.

De plus, un algorithme permet au robot de tourner la tête vers la position du joueur, afin d'éviter un effet figé. Cet ajout a été effectué tard dans le développement, lorsque nous avons remarqué que le robot ne tournait plus après l'application des textures.

Contrairement aux autres parties de l'IA, cette fonctionnalité ne nous a pas posé beaucoup de problèmes, ni d'inquiétude. À ce stade de la création du jeu, nous avions considérablement progressé dans notre maîtrise du code.

8. Graphismes

La réalisation du monde de CodeFall est plutôt difficile à imaginer. En effet, chaque salle du jeu représente une époque différente, ce qui implique qu'il peut être nécessaire de posséder plusieurs fois le même objet, mais avec un style visuel adapté à chaque période. Cela s'applique notamment aux appareils électroniques, qui évoluent énormément en peu de temps.

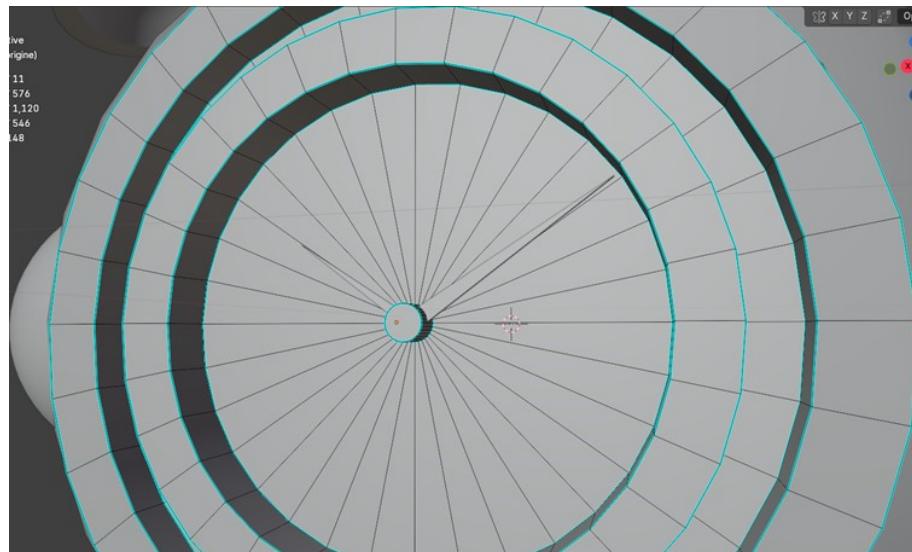
Afin de personnaliser au maximum CodeFall et de retranscrire au mieux notre vision du jeu, nous avons utilisé le logiciel de modélisation 3D Blender. Notre premier objectif a été de créer l'un des deux robots du jeu. Mais avant de le modéliser, il a d'abord fallu trouver une idée de design pour avoir une base sur laquelle s'appuyer.

Nous avons donc réfléchi aux thèmes de CodeFall, à la fonction du robot et à ce qu'il devait symboliser. Finalement, nous avons décidé que ce robot représenterait le temps. Après de nombreuses recherches sur ce thème, et en nous inspirant de créations proposées par des artistes sur des réseaux comme Pinterest, j'ai imaginé un design qui correspondait à mes attentes.

Nous avons alors commencé à le modéliser sur Blender, mais cela s'est révélé difficile, car nos connaissances étaient limitées. Nous avons donc suivi de nombreux tutoriels sur YouTube, consulté des guides et appris les raccourcis clavier utilisés fréquemment. Pour ne pas perdre ces informations, nous avons créé un dossier dans lequel nous avons retranscrit tout ce que nous utilisions afin de mieux maîtriser le logiciel.

Toutefois, retranscrire des idées en 3D peut être compliqué, car il n'existe pas d'aide universelle pour chaque cas précis. Il faut parfois repenser ses idées pour s'adapter aux contraintes du logiciel.

Lorsqu'on modélise des objets en 3D, il est essentiel de respecter certaines règles de topologie pour garantir que les modèles soient compatibles avec les moteurs de jeu, et qu'ils soient faciles à manipuler. Une règle importante consiste à utiliser uniquement des faces triangulaires ou quadrangulaires (quads).



Cette pratique est cruciale pour plusieurs raisons :

- Les moteurs comme Unity convertissent automatiquement toutes les faces en triangles pour le rendu.
- Si des polygones à plus de quatre côtés (N-gons) sont utilisés, la conversion peut produire des erreurs imprévisibles lors de l'affichage.
- Enfin, cela facilite considérablement la manipulation des objets dans les logiciels de modélisation 3D, et les triangles sont les formes géométriques les plus simples à traiter par une carte graphique.



Suite à cela, nous avons débuté la création du monde de CodeFall, celui-ci étant divisé en plusieurs salles/zones :

- La chambre
- La bibliothèque
- Le labyrinthe
- Les salles de classe

On retrouve entre les salles différents couloirs visant à faire une transition entre chaque salle.

Dans un premier temps, il a fallu trouver une méthode efficace et simple pour concevoir la structure principale de chaque salle. Initialement, nous avions envisagé une approche naïve consistant à créer un cube, l'agrandir, puis l'aplatir afin d'en faire des murs, un sol et un plafond. Cependant, nous avons rapidement constaté que cette méthode était loin d'être optimale.

En effet, cette méthode limitait considérablement les possibilités de modification et d'expansion tout en manquant de précision. Après quelques recherches, nous avons identifié ProBuilder, un outil de Unity, comme la solution idéale à ce problème.

Simple d'utilisation et intuitif, ProBuilder permet de créer une grande variété de volumes en quelques instants et offre de nombreux outils de modification. Bien que son apprentissage ait initialement ralenti l'avancement graphique du jeu, cet investissement en temps a rapidement été compensé. Désormais, la création des salles est bien plus rapide et efficace.

De plus, cet outil nous permet d'avoir un aperçu extérieur de chaque salle, ce qui facilite le placement des objets tout en offrant une immersion totale une fois à l'intérieur de la pièce.



Initialement, nous avons décidé d'utiliser uniquement des assets issus du Unity Asset Store afin de donner vie à chaque pièce. En effet, on téléchargeait des modèles 3D trouvés sur Internet, mais on rencontrait rapidement des problèmes de compatibilité. Très souvent, les objets importés ne s'affichaient pas correctement une fois placés dans la scène : soit les textures étaient absentes, soit elles ne se chargeaient pas comme prévu, laissant place à une couleur rose ou blanche, typique des matériaux manquants dans Unity.

Au début, on ne comprenait pas vraiment pourquoi cela se produisait. On pensait qu'il suffisait simplement de glisser l'objet dans la scène pour que tout fonctionne directement. On ignorait encore les étapes essentielles à respecter, comme l'importation des textures associées.

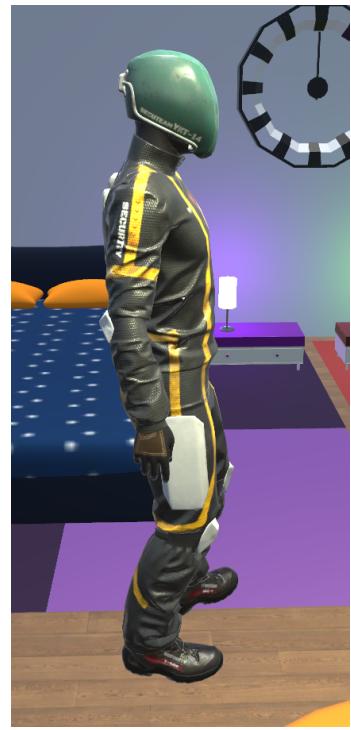
En nous renseignant davantage, on a découvert l'importance de la compatibilité avec la pipeline de rendu utilisée dans notre projet. En l'occurrence, nous travaillons avec l'URP (Universal Render Pipeline). On a donc décidé de nous concentrer uniquement sur des assets disponibles sur le Unity Asset Store, en filtrant ceux compatibles avec l'URP. Grâce à cela, les textures étaient directement appliquées aux objets dès leur importation, ce qui nous évitait bien des problèmes.

Cependant, cette solution avait ses limites. On a fini par se rendre compte que les assets disponibles gratuitement sur l'Asset Store ne suffiraient pas à couvrir l'ensemble de nos besoins graphiques, notamment lorsqu'il s'agissait de créer des ambiances très spécifiques pour certaines époques ou pièces du jeu. C'est à ce moment-là qu'on a commencé à explorer d'autres méthodes.

On a découvert qu'un matériau ne se résume pas à une simple image, mais qu'il repose sur plusieurs maps différentes : la base color (ou albedo), la normal map, l'AO (ambient occlusion), la metallic map et la roughness map. Chaque map a son utilité et permet de donner un rendu beaucoup plus réaliste et détaillé à l'objet. Ce savoir nous a permis de réutiliser des modèles provenant de sites comme Sketchfab, souvent fournis avec ces différentes textures, et de les configurer manuellement dans Unity pour qu'ils soient parfaitement intégrés dans notre scène.

En parallèle, on s'est aussi renseignés sur les formats de fichiers les plus compatibles avec Unity. On a constaté que les formats **.fbx** et **.obj** étaient les plus simples à utiliser, car ils s'importaient facilement dans Unity et étaient souvent accompagnés de leurs textures. Grâce à toutes ces connaissances, on a pu étendre notre bibliothèque d'assets et personnaliser davantage l'univers graphique de CodeFall tout en conservant une cohérence visuelle avec notre pipeline URP.

Le modèle des joueurs est un modèle trouvé sur Sketchfab. Il représente un individu masqué et mystérieux afin que chacun puisse s'identifier à celui-ci.



Il était désormais possible pour nous de créer le monde de CodeFall.

Au début de la partie, les joueurs apparaissent mystérieusement dans une chambre, où ils font la rencontre de deux robots. Le choix d'une chambre nous a semblé être le meilleur point de départ. En effet, faire apparaître les joueurs dans cette pièce permettait d'évoquer l'idée d'un début de rêve, en raison de l'atmosphère apaisante qu'elle dégage. On y retrouve des lits, des bureaux, des étagères, afin qu'ils puissent y trouver tout ce dont ils ont besoin.



Après avoir résolu la première énigme, les joueurs apparaissent dans un couloir d'hôpital. L'univers médical a ici été évoqué pour deux raisons :

- Perturber les joueurs qui viennent d'ouvrir la porte d'une simple chambre ;
- Évoquer l'idée du temps.

En effet, les couloirs d'hôpital peuvent être synonymes d'angoisse, de tristesse ou de joie, selon la situation de chacun. C'est donc un lieu chargé d'émotions et de confusion. De plus, le temps semble parfois s'y écouter différemment, en raison de l'incertitude quant à ce qui se trouve derrière les portes, et à cause de l'absence de fenêtres, empêchant toute perception du monde extérieur.

Il y a deux portes dans cette pièce, mais elles sont malheureusement impossibles à ouvrir pour les deux joueurs. À la fin du couloir, les joueurs aperçoivent une vieille horloge, ainsi qu'un tableau représentant une bibliothèque, introduisant ainsi la salle suivante.



Cette nouvelle salle, la bibliothèque, est mystérieuse, car elle semble ancienne, en raison de son apparence et de l'utilisation de torches plutôt que d'ampoules électriques pour l'éclairage. Toutefois, on y trouve des portes à code, et les étagères sont disposées de manière étrange. En effet, les joueurs devront découvrir que certaines de ces étagères forment des chiffres, ce qui leur permettra de résoudre les énigmes de la pièce.



Une fois sortis de cette salle, les joueurs se retrouvent dans le même couloir qu'auparavant, mais totalement désordonné. Leurs actions ont des répercussions visibles sur le monde. Ce couloir, qui semblait auparavant normal, contient désormais des bancs de salle d'attente flottant dans les airs, des problèmes de textures, ainsi que des objets dont la taille a été modifiée.



Par la suite, les joueurs se retrouvent séparés, chacun dans un labyrinthe semblant situé dans une ville. En réalité, les deux labyrinthes sont identiques, à quelques détails près. Les murs sont en briques, le sol est une route, et de grands immeubles les entourent. On y retrouve divers éléments caractéristiques d'un environnement urbain : panneaux de signalisation, arrêts de bus, ainsi que d'autres objets placés là de manière mystérieuse, comme des étagères issues de la première salle.



Une fois encore, après avoir résolu les énigmes du labyrinthe, les joueurs reviennent dans le couloir. Cette fois, plus rien ne semble avoir de sens. Le couloir est à l'envers, les objets ne sont plus à leur place, certains sont étirés, mais ils forment, de manière étrange, des escaliers permettant aux joueurs d'accéder aux portes.

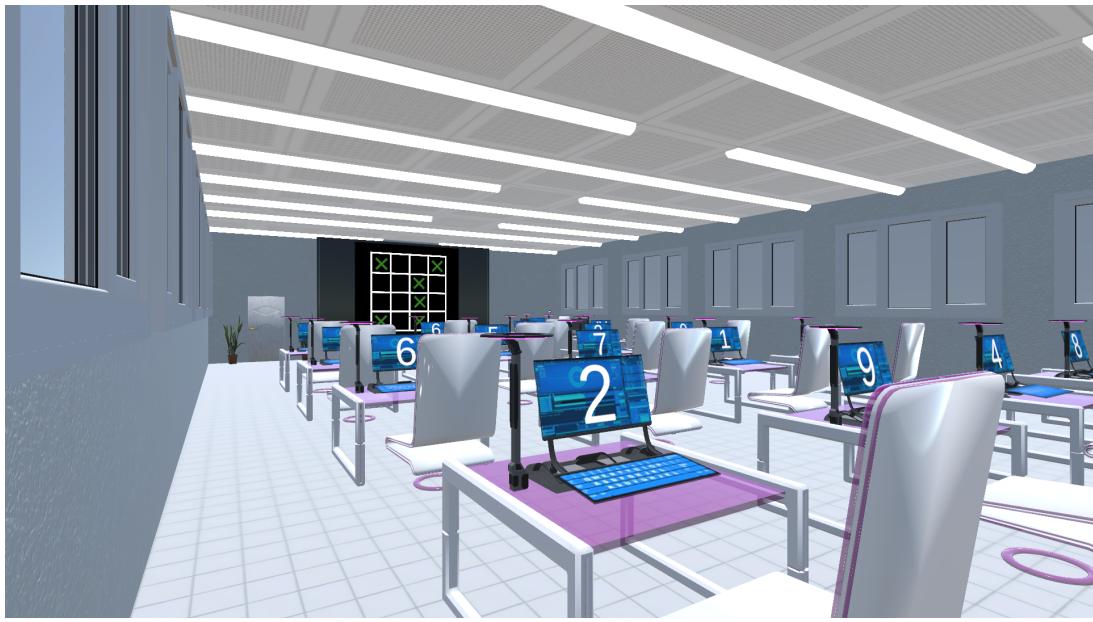


Derrière ces fameuses portes, les joueurs découvrent ce qui ressemble à des salles de classe futuristes. Ils prennent alors conscience qu'ils semblent effectuer une sorte de voyage dans le temps. En effet :

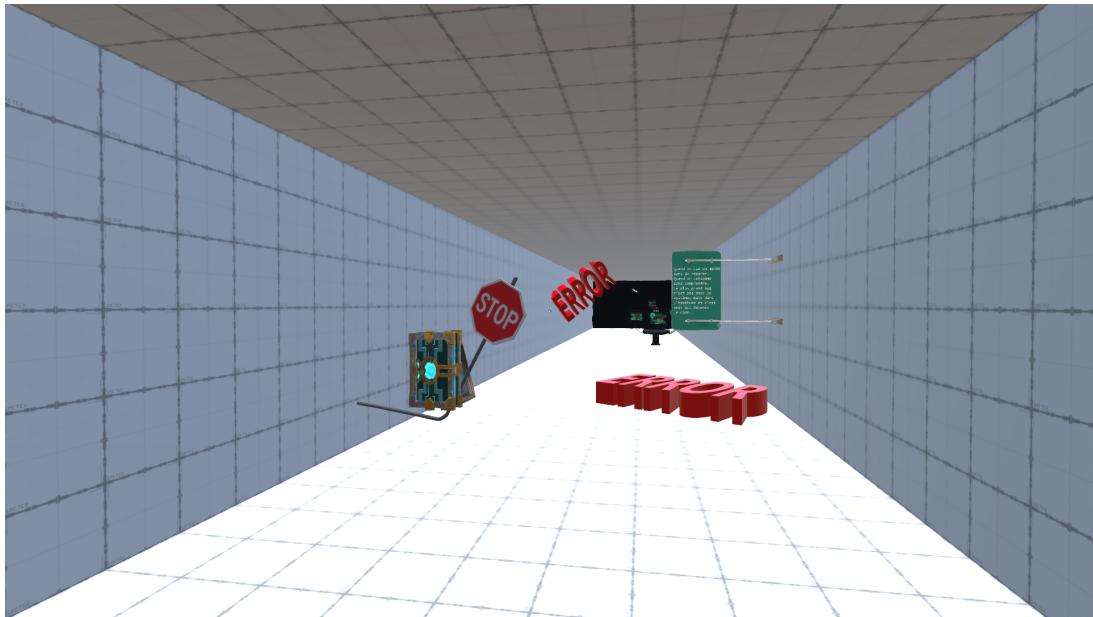
La bibliothèque représente le passé ;

Le labyrinthe, le présent ;

Les salles de classe, le futur.



Enfin, une fois la dernière porte ouverte, un long couloir presque vide et dénué de textures les attend. Des objets flottants, provenant de toutes les époques, y sont dispersés, comportant des messages adressés aux joueurs. À la fin de cette dernière pièce, seul un grand vide les attend, comme si le monde avait disparu.



9. Animations

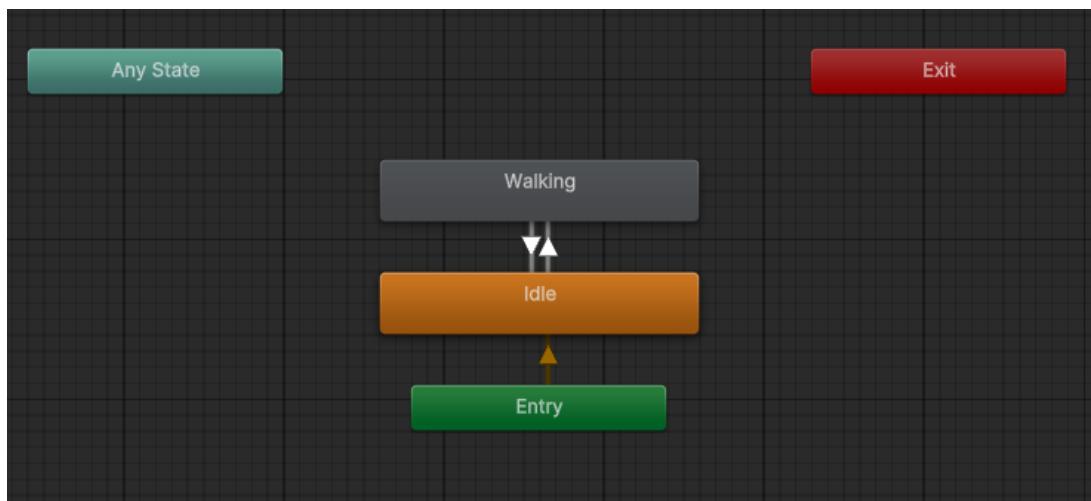
A. Personnage

Afin de donner vie aux personnages des joueurs, nous leur avons attribué des animations. Pour cela, nous avons utilisé Mixamo, un site en ligne qui propose gratuitement un large catalogue de personnages 3D ainsi que des animations variées. Nous avons sélectionné deux animations essentielles pour notre prototype :

- une animation dite idle, correspondant à la posture du personnage lorsqu'il est immobile ;
- une animation de marche (walking), utilisée lorsqu'il se déplace.

Une fois ces animations importées dans Unity, il ne suffisait pas de les attacher directement au modèle. Il a fallu les organiser dans un Animator Controller, afin de gérer dynamiquement les transitions entre les différents états d'animation. Pour cela, nous avons créé un graphe d'états simple, composé de deux nœuds principaux : Idle et Walking. Le personnage commence toujours dans l'état Idle et peut ensuite passer à l'état Walking, ou y revenir, selon qu'il bouge ou non.

Ces transitions sont contrôlées grâce à un paramètre booléen nommé isWalking, que nous avons défini dans l'Animator. Ce paramètre est modifié dynamiquement dans un script nommé PlayerAnim.



Ce script détecte les touches pressées par le joueur (Z, Q, S, D) afin de déterminer s'il est en train de marcher ou non. Si l'une de ces touches est enfoncée, le paramètre isWalking passe à true, ce qui déclenche la transition vers l'animation Walking. Dès que le joueur relâche les touches, l'animation revient à Idle. Cela permet de rendre le comportement du personnage réactif et fluide, en adaptant automatiquement son animation à ses actions.

Ce système, bien que simple, nous a permis de poser les bases de l'animation de notre personnage tout en assurant une intégration propre avec les contrôles du joueur.

B. Robots

Nos robots, conçus comme des compagnons du joueur dans l'univers du jeu, ont également bénéficié d'un travail d'animation visant à renforcer leur caractère vivant et dynamique. Pour chacun d'eux, nous avons mis en place une animation de flottement, qui donne l'illusion qu'ils lévitent doucement dans les airs.

Cette animation a été réalisée directement dans Unity, à l'aide de la fenêtre Animation. En modifiant les coordonnées verticales du robot (axe Y), nous avons créé un mouvement de montée et de descente lent et régulier, se répétant en boucle. Ce mouvement constant donne au robot un effet de légèreté et renforce l'illusion qu'il flotte dans l'espace.

Afin de rendre les anneaux du robot "Bop" encore plus réalistes, nous lui avons appliqué une rotation continue autour de l'axe vertical. Cette rotation est gérée automatiquement par un script attaché à l'objet de l'anneau : ce script fait tourner l'anneau sur lui-même en permanence, à une vitesse constante. L'effet produit est subtil, mais très efficace pour donner un aspect dynamique au robot.

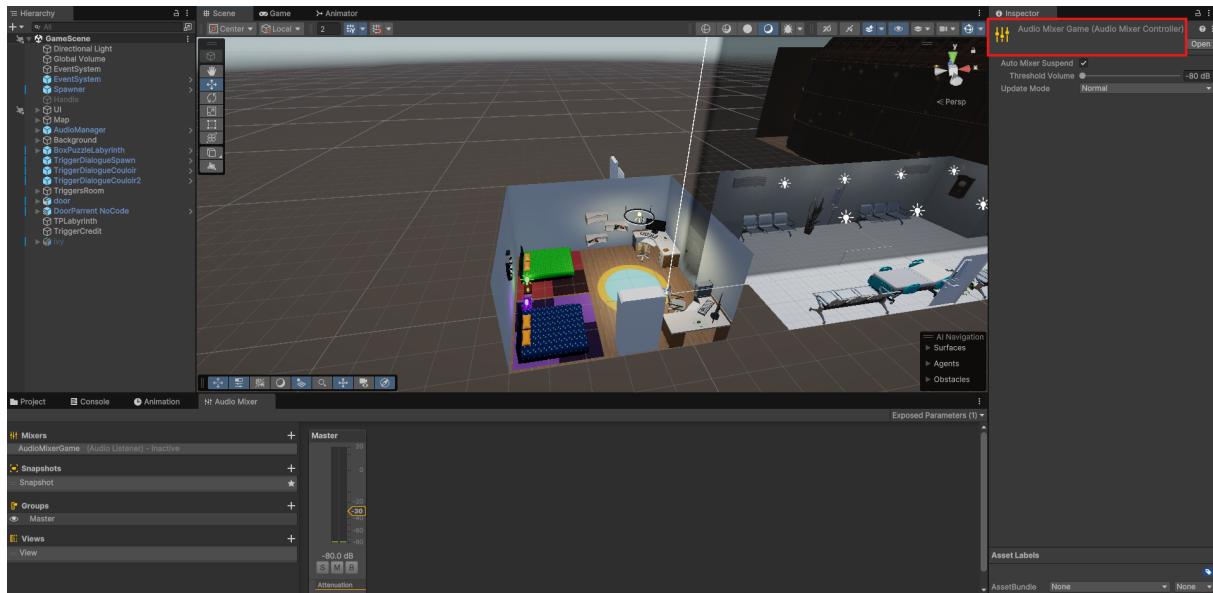
Ces animations, bien que simples, jouent un rôle essentiel dans la personnalité de nos robots et contribuent à enrichir l'univers visuel du jeu.

10. Musiques et sons

L'objectif pour l'audio était d'avoir une musique de fond, de pouvoir changer son volume, de faire varier la musique selon la salle, et d'ajouter des bruitages.

Pour la musique, nous avons choisi des morceaux d'ambiance calmes et mystérieux, afin que le joueur puisse rester concentré sur la résolution des énigmes.

Pour permettre la modification du volume, nous avons ajouté un Audio Mixer, qui, combiné avec le script des paramètres, permet de modifier le volume du jeu en direct.



De plus, nous avons ajouté des bruitages dans le jeu, par exemple des sons de robots pendant les dialogues, ou des bruits de "glitch" lorsqu'un joueur est téléporté.

Le script fait en sorte que, lorsqu'une lettre s'affiche à l'écran, un bruitage est joué, et qu'à la fin du dialogue, le son s'arrête automatiquement. Pour changer la musique en fonction de la salle, nous avons créé un box collider invisible qui arrête la musique en cours et lance une nouvelle piste en boucle. Une fois cela fait, le script se désactive lui-même afin d'éviter que la musique ne change en continu.

Nous avons sélectionné quatre musiques différentes pour notre jeu. Nous avons choisi d'utiliser des morceaux de musique classique, afin de créer une ambiance à la fois sympathique et dynamique. Voici les morceaux utilisés :

- "Ma Mère l'Oye II. Petit Poucet" de Maurice Ravel : utilisée pour la bibliothèque et le labyrinthe, car elle est très calme et propice à la réflexion.
- "Hungarian Rhapsody No. 2" de Franz Liszt (piano) : utilisée à la fin du jeu, car elle est puissante et dramatique, marquant le chaos final.
- "Prelude to « Lohengrin »" de Richard Wagner : utilisée dans le menu d'attente, pour son ambiance calme et magique.

- “La grenouille” de Florent Fossati : utilisée pour le spawn, les crédits et le menu principal. Cette musique a été composée directement par un membre de notre groupe, en charge de l’aspect sonore. Il s’agit d’un duo flûte-piano qui crée une ambiance joyeuse, parfaite pour introduire le jeu et donner envie de jouer.

L’ambiance sonore du jeu n’a pas été négligée. Nous avons passé du temps à sélectionner les musiques avec soin.

L’implémentation de la musique dans le jeu ne nous a pas posé de problème au départ. Cependant, nous avons rencontré des difficultés pour faire fonctionner le changement dynamique de musique en cours de partie. Malgré tout, nous avons surmonté ces obstacles et réussi à créer une ambiance sonore agréable et immersive.

III. Bilan sur l'avancement

Tâches	Prévu	Réalisé
histoire et dialogues	100%	100%
Graphismes/Texture	100%	100%
Enigmes et Mécaniques de jeu	100%	100%
Multijoueur	100%	100%
Menu/Options	100%	100%
Site Web	100%	100%
IA	100%	100%
Audio	100%	100%

IV. Conclusion

Le développement de CodeFall a représenté bien plus qu'un simple projet technique : c'est avant tout une aventure humaine qui a été profondément enrichissante. Concevoir ce jeu nous a appris à collaborer de manière efficace, à faire face aux imprévus, mais aussi à donner vie à une vision partagée et ce en mêlant narration, gameplay et engagement.

Travailler en équipe nous a permis de confronter nos idées, d'élargir nos compétences et d'apprendre les uns des autres. Que ce soit pour le développement du mode multijoueur, le scénario, la création des mécaniques d'éénigmes ou le travail sur le design sonore et visuel, chaque étape a été l'occasion de progresser sur le plan technique, tout en restant cohérents fidèles à l'univers et aux messages que nous voulions transmettre.

CodeFall n'est pas seulement un jeu : c'est aussi une réflexion sur notre rapport au monde, à la technologie et aux conséquences de nos choix. En intégrant volontairement des « bugs » comme éléments de gameplay, nous avons voulu mettre en évidence la frontière entre progrès et destruction, entre confort et responsabilité. Le joueur, en interagissant avec un monde instable, devient à la fois acteur de son évolution et témoin de ses dérives.

Même si le projet reste perfectible, nous sommes très fiers du chemin parcouru. Ce que nous retiendrons surtout, ce sont les heures passées à chercher, corriger, discuter, tester et créer ensemble. Ce jeu est le fruit d'un engagement sincère, et peut-être même le point de départ d'une longue série de projets encore meilleurs.