

Rapport de soutenance n°1

Janvier 2025

BLACKOUT

Prénoms et noms des membres :

Wassim ALOUINI

Brewen MERER

Loan MARIA-CHATELAIN

Fabien GUESSANT

Elliott MIEZE

Groupe : **Winter-Architect**

EPITA Promo 2029

Sommaire détaillé

1	Introduction	4
1.1	Répartition des tâches	4
2	Avancement	4
2.1	Multijoueur	4
2.1.1	Choix de la Solution Technique	4
2.1.2	Modèle Host-Client	5
2.1.3	Problèmes et Tests à Réaliser	5
2.2	Interfaces graphiques.	5
2.2.1	Gestion des Paramètres.	5
2.2.2	Interface pour les Énigmes.	5
2.2.3	Technologie Utilisée.	6
2.2.4	Problèmes rencontrés.	6
2.2.5	Améliorations Futures.	6
2.3	Intelligence artificielle	7
2.3.1	Objectif de L'IA.	7
2.3.2	Utilisation du Package AI Navigation.	7
2.3.3	Déplacement Automatique.	8
2.3.4	Détection du Joueur.	8
2.3.5	Gestion de la Recherche.	8
2.3.6	Gestion des États de l'IA.	8
2.3.7	Point de vue sur la situation actuelle	9
2.4	Level design	9
2.4.1	IA ennemis	9
2.4.2	Composition des salles	10
2.4.3	Travail de recherche et problématiques	10
2.4.4	Exemples d'anomalies et d'énigmes	10
2.4.5	Ajouts futurs	12
2.5	Site web	12
2.5.1	Structure du Site.	12
2.5.2	Boutons de navigation.	12
2.6	Technologies Utilisées	13
2.6.1	Hébergement Actuel et Futur	13
2.6.2	Mouvements et interactions	13
2.6.3	Contrôles des Joueurs.	13
2.6.4	Difficultés rencontrées.	15
2.6.5	Progrès et Objectifs Restants.	15
2.7	Modélisation 3D	15

2.7.1	Outil Utilisé.	15
2.7.2	Méthode de Construction des Assets.	16
2.7.3	Avantages de la Méthode.	16
2.7.4	Processus d'Exportation.	16
2.7.5	Problèmes Rencontrés.	17
2.7.6	Objectifs pour la Prochaine Soutenance.	17
2.7.7	Exemples	17
3	Conclusion	18
3.1	Bilan sur l'avancement	18
3.2	Perspectives et futur du projet	19
3.3	Réflexion et apprentissages	19

1 Introduction

Ce rapport est le premier rapport de soutenance du jeu Blackout. Il a pour objectif de présenter l'avancée du projet.

1.1 Répartition des tâches

Référent	Suppléant	Tâche
Fabien	Wassim	Scénario
Loan	Brewen	Gameplay
Eliott	Wassim	Interfaces graphiques
Fabien	Eliott	Modélisation 3D
Wassim	Loan	Animation
Eliott	Wassim	Site Web
Brewen	Fabien	IA
Loan	Eliott	Level design

Table 1: Repartition des tâches

2 Avancement

2.1 Multijoueur

2.1.1 Choix de la Solution Technique

Afin de nous éviter de multiples problèmes dans la suite du développement, nous avons décidé de commencer par le développement du **multijoueur**, ce dernier représentant une part importante et nécessaire du projet.

Pour cela, nous avons choisi d'utiliser la solution **NetCode for GameObjects (NGO)**, un framework multijoueur intégré à **Unity**, qui simplifie la création et la gestion des jeux en réseau. Cette solution repose sur une architecture serveur-client et offre des fonctionnalités clés telles que :

- La **synchronicité automatique** des objets réseau,
- La prise en charge des **Remote Procedure Calls (RPC)** pour exécuter des actions à distance,
- Une gestion optimisée des **rôles et de l'autorité** entre le serveur et les clients.

En choisissant NGO, nous bénéficions d'une solution robuste, évolutive et parfaitement adaptée à l'écosystème Unity, ce qui nous permet d'assurer une meilleure stabilité et une intégration fluide des fonctionnalités multijoueurs dès les premières étapes de développement.

2.1.2 Modèle Host-Client

En complément, pour minimiser les coûts liés à l'infrastructure réseau, nous avons décidé d'utiliser le modèle **host-client**. Cela signifie que ce sont les joueurs eux-mêmes qui feront office de serveur. Concrètement, un premier joueur crée le "**serveur**" puis génère un code de connexion à communiquer à l'autre joueur afin qu'il puisse se rejoindre dans le lobby. Par exemple, un joueur A crée une partie en tant qu'hôte, génère un code de connexion et le communique au joueur B qui peut ainsi rejoindre le lobby via l'interface dédiée.

2.1.3 Problèmes et Tests à Réaliser

Bien que nous ayons réalisé une grande partie du développement multijoueur, chaque nouvelle fonctionnalité doit être testée en conditions multijoueurs. Nous devons également effectuer de nombreux tests, notamment :

- Tester le comportement lorsque le joueur hôte se déconnecte,
- Vérifier les performances du jeu en fonction de la qualité du réseau de chaque joueur.

2.2 Interfaces graphiques.

Nous avons développé plusieurs interfaces graphiques essentielles pour le jeu. Tout d'abord, un **menu** permet au joueur de créer une partie ou de rejoindre un ami dans sa partie. Ce menu conduit ensuite à un autre écran, clé du jeu, qui permet de lancer la partie une fois que les deux joueurs sont prêts.

2.2.1 Gestion des Paramètres.

De plus, une **interface de paramètres** permet au joueur de gérer les options audio, comme le volume, ainsi que de modifier la qualité des graphismes.

2.2.2 Interface pour les Énigmes.

Pour accompagner les énigmes du jeu, nous avons conçu diverses interfaces, notamment un **keypad** permettant au joueur de saisir des codes pour résoudre les énigmes.

2.2.3 Technologie Utilisée.

Ces différentes interfaces ont été réalisées à l'aide de l'**UI Toolkit** d'Unity. Cet outil nous permet de créer des interfaces plus avancées et de manière plus simple qu'avec le système classique de canvas.

2.2.4 Problèmes rencontrés.

Cependant, le UI Toolkit a présenté certaines difficultés, notamment en ce qui concerne le côté **responsive**, qui entraînait parfois des bugs dans l'affichage des interfaces.

2.2.5 Améliorations Futures.

Dans le futur, nous prévoyons d'implémenter davantage d'interfaces, notamment pour les énigmes. Parmi les ajouts envisagés :

- Un **terminal interactif** sur lequel les joueurs pourront voir et écrire du texte, ainsi qu'interagir avec des boutons.
- L'extension des **paramètres** avec la possibilité de modifier les contrôles du jeu.
- La création d'un **masque spécifique** visible uniquement par le support, donnant l'illusion de voir à travers une caméra.
- L'ajout d'une fonctionnalité dans le lobby permettant de **sélectionner son rôle**, car actuellement, le premier connecté est l'agent et le second le support.

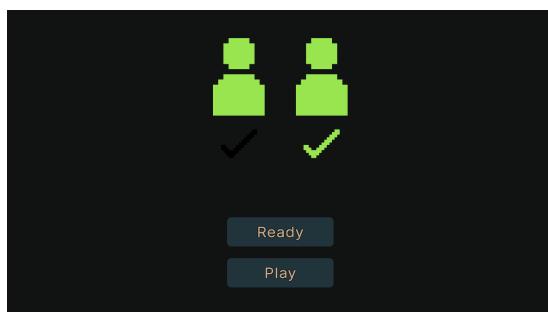


Figure 1: Lobby

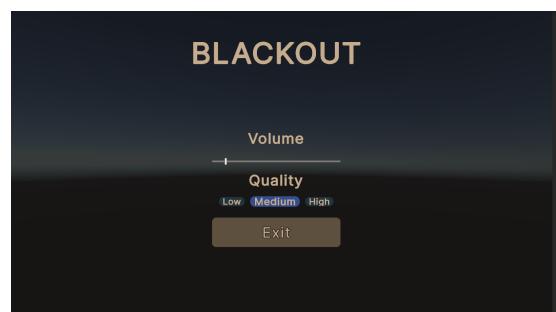


Figure 2: Menu Paramètres

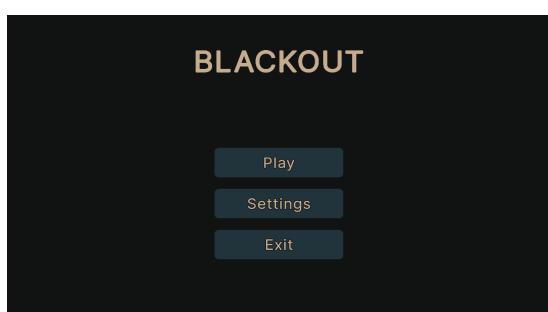


Figure 3: Menu principal

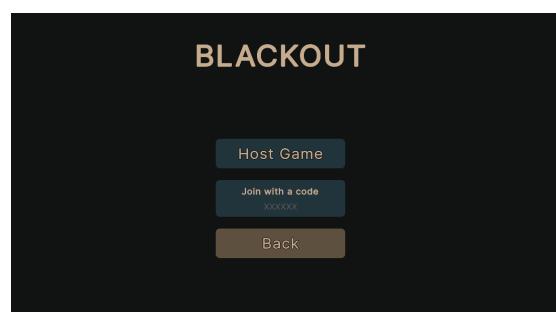


Figure 4: Menu de lancement du jeu



Figure 5: Keypad.

2.3 Intelligence artificielle

2.3.1 Objectif de L'IA.

Le but de nos intelligences artificielles est de donner une complexité aux joueurs qui va au-delà de l'intellect nécessaire aux énigmes. Nous souhaitons que notre jeu ne se résume pas seulement à un enchaînement de puzzles à résoudre, mais aussi à quelque chose de stimulant en alternant énigmes, parcours et fuites.

Combiner épreuves de réflexion et épreuves de traque permettrait d'offrir des situations plus effrayantes, angoissantes et stressantes, nuisant à la capacité de réflexion des joueurs. Cela aurait pour effet de rendre la résolution de problèmes plus ou moins simples plus difficile.

Une telle dynamique immergerait les joueurs dans une situation où l'hésitation peut s'avérer fatale, provoquant alors plus d'émotions que dans des situations où ils ne risquent rien.

2.3.2 Utilisation du Package AI Navigation.

Concernant l'implémentation de l'intelligence artificielle, nous nous servons du package **AI Navigation** qui permet l'implémentation d'un système de *pathfinding* de manière efficace. Ceci est réalisé en construisant (à l'aide de ce package) une surface servant à déterminer où peuvent se déplacer les IA ennemis.

Il a fallu comprendre le fonctionnement de ce package, avec le principe de collisions entre les intelligences artificielles et les murs, sachant que notre IA pouvait parfois penser pouvoir se déplacer entre deux obstacles lorsque nos collisions étaient mal gérées.

2.3.3 Déplacement Automatique.

Nous avons ajouté à l'IA la fonctionnalité de se déplacer dans l'espace sans qu'on ne le lui demande. Cela signifie que nous avons imposé à l'IA de suivre un *pattern* spécifique en évitant les obstacles. Ce *pattern* est défini à l'aide de nœuds placés dans l'environnement qui servent d'étapes de déplacement.

2.3.4 Détection du Joueur.

Pour donner un sens au déplacement, il nous fallait détecter le joueur lors du déplacement de l'IA. Nous avons tout d'abord créé une zone de collision invisible entourant notre IA pour repérer ce joueur lorsqu'il y entre. Cependant, cette méthode présentait des problèmes :

- L'IA détectait le joueur même de dos.
- L'IA ignorait les murs lors de la traque du joueur.

Pour corriger cela, nous avons repensé la détection en :

1. Crément un champ de vision délimité par des vecteurs.
2. Utilisant des *raycasts* pour détecter la présence du joueur tout en tenant compte des obstacles.

2.3.5 Gestion de la Recherche.

Lorsque le joueur n'est plus dans le champ de vision de l'IA, celle-ci se déplace à la dernière position où elle l'a détecté. Si le joueur reste introuvable, l'IA reprend son *pattern*. Une amélioration envisageable serait d'ajouter un comportement où l'IA cherche activement autour d'elle avant de reprendre son *pattern*, rendant le jeu plus réaliste et immersif.

2.3.6 Gestion des États de l'IA.

Pour une meilleure gestion, il est essentiel de définir plusieurs états pour l'IA, comme :

- **Recherche** : L'IA explore l'environnement à la recherche du joueur.
- **Traque** : L'IA poursuit activement le joueur.
- **Attaque** : L'IA engage une interaction directe avec le joueur.

Des états supplémentaires spécifiques à certaines IA pourraient être ajoutés, comme :

- **Cache** : L'IA se cache pour surprendre le joueur.
- **Stalk** : L'IA suit le joueur discrètement pour créer une ambiance de suspense.

Ces états permettent de structurer le code de manière modulaire, facilitant ainsi la gestion et l'évolution de l'IA.

2.3.7 Point de vue sur la situation actuelle

Les fonctionnalités actuelles offrent une base solide pour une IA immersive, mais des améliorations sont envisageables pour renforcer l'expérience de jeu et le réalisme.

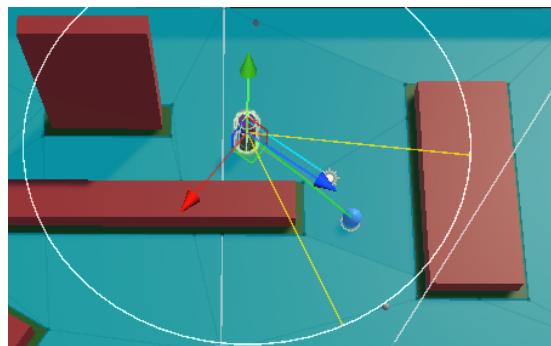


Figure 6: IA poursuivant le joueur

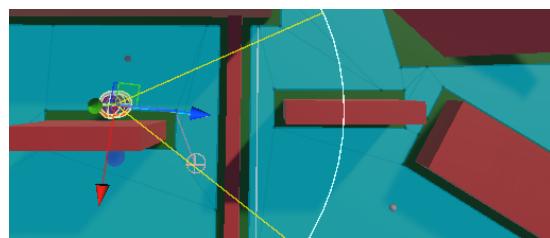


Figure 7: IA se déplaçant vers la dernière position du joueur aperçue

Légende :

- **Bleu** : Zone accessible à l'IA.
- **Rouge** : Prochaine destination.
- **Jaune** : Délimitation du cône de vision.
- **Vert** : Raycast touchant le joueur.

2.4 Level design

2.4.1 IA ennemis

Concernant le *Level Design*, il nous a fallu travailler sur deux domaines majeurs : dans un premier temps, les **IA ennemis** : quel est leur **comportement** dans une salle ? Comment **réagissent-elles face à un joueur** ? Quelles sont leurs **fonctionnalités** ?

2.4.2 Composition des salles

Dans un second temps, la **composition des salles** : quels **ennemis** peut-on trouver dans chacune d'entre elles ? Quelles sont les **énigmes** auxquelles le joueur devra faire face ?

2.4.3 Travail de recherche et problématiques

Afin de traiter chacune de ces problématiques, il a été primordial d'effectuer un **travail de recherche** au préalable. Nous avons commencé par représenter **schématiquement** les différentes salles, énigmes et anomalies auxquelles le joueur sera confronté.

Plusieurs problèmes se sont alors posés :

- *Concernant les énigmes* : il faut s'assurer qu'elles soient toutes **compréhensibles facilement**. Il est primordial que les joueurs puissent **facilement comprendre ce que l'on attend d'eux**, sans pour autant que la solution de l'énigme ne soit trouvable trop facilement.
- *Rôle de l'Agent et du Support* : comment s'assurer que le rôle de l'**Agent** soit tout aussi important que le rôle du **Support** ? À première vue, le rôle d'Agent semble bien plus **attrayant** que celui du Support, étant donné que c'est le seul à pouvoir **errer dans les salles et progresser**, tandis que le Support ne peut qu'**observer**. Pourtant, le Support joue un **rôle vital**. Il faut donc prendre soin d'ajuster chaque élément du jeu pour que les deux joueurs puissent avoir **le même impact** sur la progression du jeu.
- *Diversité des salles* : comment faire pour que chaque **salle se démarque clairement de la précédente** ? Une de nos plus grandes craintes serait que les joueurs **s'ennuient**. Si les salles, énigmes et IA sont trop **redondantes**, alors les joueurs risquent fortement de se **lasser**, et l'intérêt de continuer à jouer serait moindre. Par conséquent, chaque élément du jeu doit se **démarquer le plus possible** des autres.

2.4.4 Exemples d'anomalies et d'énigmes

Exemple d'anomalie : Voici un exemple d'anomalie (IA ennemie) à laquelle les joueurs devront se confronter :

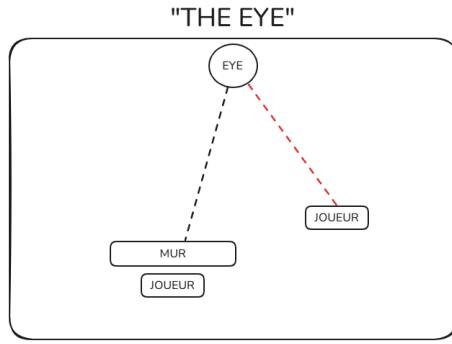


Figure 8: anomalie ”The Eye”

Son fonctionnement est simple : dès que le joueur se situe dans son champ de vision, elle lui inflige des **dégâts en continu**. Afin de lui échapper, il suffit de se **cacher derrière des éléments du décor**, comme un mur, par exemple.

Comme mentionné précédemment, il nous a fallu réfléchir à la façon dont le **Support** pourrait se rendre utile. Les solutions sont nombreuses :

- Le Support pourrait avoir la possibilité de **bouger des éléments du décor** pour bloquer la vue de l'anomalie.
- Le Support pourrait être capable de **neutraliser l'anomalie pendant un certain laps de temps**, permettant ainsi à l'Agent de progresser vers son objectif.

Exemple d'énigme : Voici un deuxième exemple, cette fois-ci d'une **énigme** :

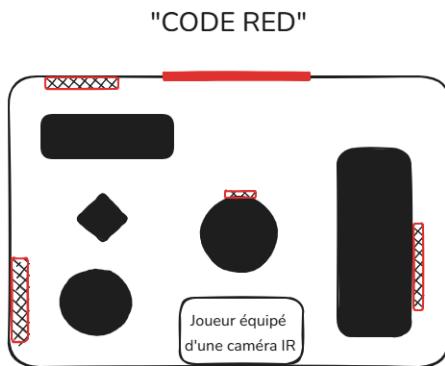


Figure 9: Énigme ”Code Red”

Légende :

- **Noir** : Eléments de la salle.
- **Rouge** et **Noir** : Morceaux de code.
- **Rouge** : Porte de sortie.

Le fonctionnement est le suivant : l'**Agent** est équipé d'une **caméra infrarouge**, permettant au Support de **suivre sa progression**. À travers cette caméra, le Support peut voir des éléments qui ne sont **pas visibles aux yeux de l'Agent**, comme des **morceaux de code écrits aux murs**.

Pour rendre la tâche plus complexe, il serait possible d'ajouter des **anomalies** dans la salle, poursuivant l'Agent et rendant l'exploration plus difficile.

2.4.5 Ajouts futurs

À l'avenir, nous souhaitons :

- Ajouter davantage d'**énigmes et d'anomalies**.
- Mettre l'accent sur le **côté exploration de l'environnement**. Par exemple :
 - Certaines salles pourraient être **beaucoup plus grandes**.
 - D'autres pourraient avoir **plusieurs étages**.
 - Certaines salles pourraient être **séparées en plusieurs parties**.

2.5 Site web

2.5.1 Structure du Site.

La page d'accueil du site est simple et présente une brève description de l'entreprise, accompagnée de quelques boutons permettant d'explorer le reste du site.

2.5.2 Boutons de navigation.

Les principaux boutons permettent d'accéder aux sections suivantes :

- **Membres et réseaux sociaux** : Le premier bouton dirige vers une page présentant nos membres et leurs réseaux sociaux.
- **Jeux développés** : Le second bouton mène à une section dédiée aux jeux développés par notre studio, dont le seul jeu actuellement disponible, *Blackout*. Cette page contient :
 - une présentation du jeu,
 - un lien pour le télécharger,
 - des documents utiles,
 - des statistiques,
 - et une *timeline* retracant le développement.
- **Présentation de l'entreprise** : La dernière page met en lumière l'entreprise, son origine, et ses valeurs fondamentales.

2.6 Technologies Utilisées

Le site a été développé avec des technologies modernes, choisies pour leur efficacité et leur flexibilité :

- **NuxtJS (avec VueJS)** : Utilisé pour gérer à la fois le front-end et le back-end.
- **TailwindCSS** : Employé pour concevoir le design du front-end.

Ces technologies ont été choisies en raison de l'expérience préalable des membres responsables du site et de leur capacité à soutenir des fonctionnalités avancées à l'avenir.

2.6.1 Hébergement Actuel et Futur

Actuellement, le site est hébergé sur **Netlify**, une solution gratuite qui répond parfaitement à nos besoins actuels. Cependant, à terme, nous prévoyons de migrer vers un **VPS sous Ubuntu**, offrant ainsi une plus grande flexibilité, notamment pour l'implémentation de statistiques dynamiques.

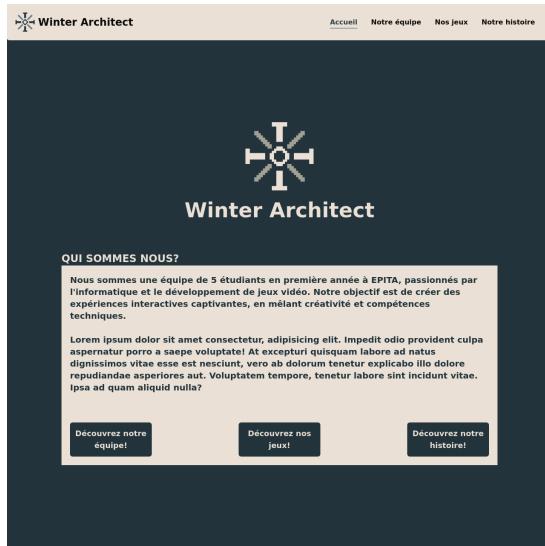


Figure 10: Page d'accueil du site

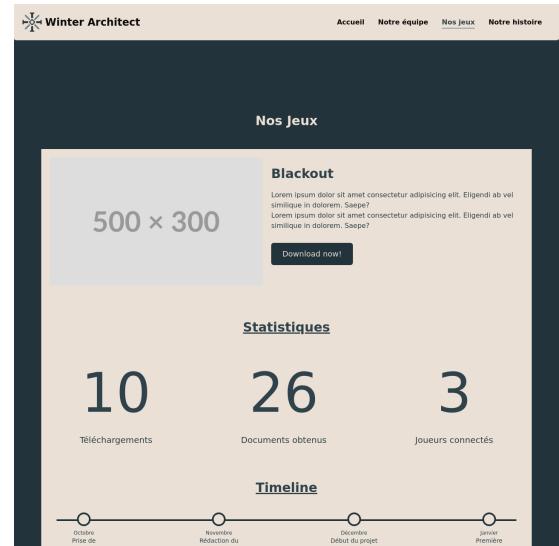


Figure 11: Page du jeu sur le site

2.6.2 Mouvements et interactions

2.6.3 Contrôles des Joueurs.

Chaque joueur possède des **mouvements uniques**. Ainsi, nous avons deux types de joueurs différents avec des contrôles différents, donc deux **scripts** différents : un pour **l'agent** et un pour **le support**.

Ainsi, **l'agent** possède les contrôles classiques d'un jeu :

- [Z] [Q] [S] [D] [Espace] pour se déplacer dans l'environnement

- **shift ↑** pour accélérer
- **mb1** plus des mouvements de souris pour tourner la caméra

Nous avons donc séparé ces contrôles en deux fonctions différentes, ce qui nous permet une gestion des erreurs plus efficace ainsi qu'une meilleure compréhension.

Voici donc une version simplifiée des méthodes de la classe Agent :

```

1   void ControlCamera()
2   {
3
4       xRotation += yMouseInput * yMouseSensitivity;
5       xRotation = Mathf.Clamp(xRotation, -90f, 90f);
6
7       yRotation += xMouseInput * xMouseSensitivity;
8
9       Quaternion verticalRotation = Quaternion.Euler(
10          xRotation, yRotation, 0f);
11
12      playerCameraPivotTransform.localRotation =
13          verticalRotation;
14
15
16      void Move()
17      {
18          Vector3 myForward = new Vector3(playerCamera.forward.x
19              , 0, playerCamera.forward.z).normalized;
20          Vector3 myRight = new Vector3(playerCamera.right.x,
21              0, playerCamera.right.z).normalized;
22
23          Vector3 myMovement = (myForward * yInput + myRight *
24              xInput).normalized * speed;
25          Vector3 myVelocity = new Vector3(myMovement.x,
26              playerRigidbody.linearVelocity.y, myMovement.z);
27          playerBody.LookAt(transform.position + myMovement);
28          playerRigidbody.linearVelocity = myVelocity;
29
30
31          isMoving = xInput != 0 || yInput != 0;
32          Animate();
33      }

```

Le support, quant à lui, utilise pour l'instant ces contrôles :

- **[mb1]** changer de caméra (il doit en pointer une autre)
- Tourner la caméra avec des mouvements de souris

2.6.4 Difficultés rencontrées.

Nous avons rencontré une difficulté principale lors de l'implémentation des **caméras**, car nous voulions que cette dernière bouge en fonction de la vue du **support**, ce qui fait que nous devions aussi gérer le côté **multijoueur** de cette caméra et trouver un moyen pour elle de bouger en fonction d'un certain **point fixe** (point de fixation de la caméra).

Voici donc une version simplifiée de la classe Support

```
1 public class Support : NetworkBehaviour
2 {
3     void Update()
4     {
5         if (!IsOwner){
6             return;
7         }
8         //Move Camera
9         myCamera.transform.Rotate(new Vector3(0, Input.GetAxis
10            ("MouseX"), 0));
11        myCamera.transform.localRotation = Quaternion.Euler(0,
12            myCamera.transform.rotation.eulerAngles.y, Input.
13            GetAxis("MouseY"));
14    }
15 }
```

2.6.5 Progrès et Objectifs Restants.

Il nous reste encore de nombreux **contrôles** à implémenter afin d'atteindre l'objectif que nous avions prévu, notamment **l'interaction avec les objets** présents dans la scène ainsi que les **électroniques** que le support contrôlera.

2.7 Modélisation 3D

2.7.1 Outil Utilisé.

Pour la modélisation 3D des objets, nous utilisons **Roblox Studio**. Ce choix s'explique par le fait que nos membres utilisaient déjà cet outil avant de commencer le projet, ce qui leur permet d'être plus productifs dans la création des assets.

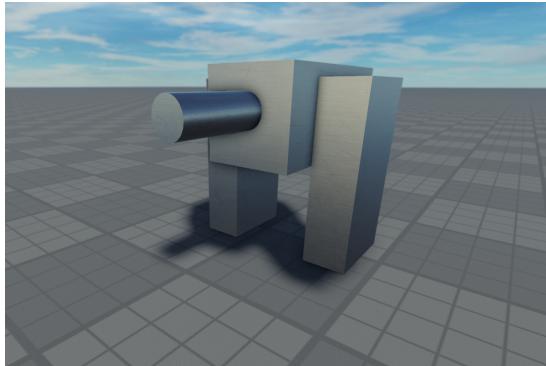
2.7.2 Méthode de Construction des Assets.

La conception des assets, qu'ils soient environnementaux (meubles, salles) ou qu'ils concernent l'apparence visuelle 3D de certains éléments utilisables, suit une méthode en plusieurs étapes :

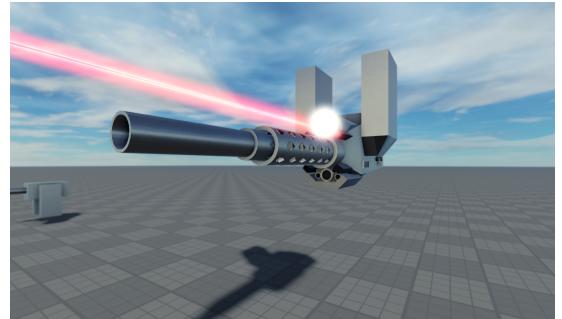
1. **Construction de la forme globale** : Nous commençons par assembler des formes simples pour obtenir une idée générale de la forme de l'objet.
2. **Affinage et précision** : Une fois la forme globale obtenue, l'affinons et ajoutons des détails intermédiaires.
3. **Ajout des petits détails** : Les plus petits détails sont ajoutés, tels que les spécificités visuelles ou les éléments décoratifs pour un environnement plus immersif.
4. **Définition des particularités** : Enfin, nous déterminons les caractéristiques spécifiques de l'objet, par exemple les parties qui doivent pouvoir se mouvoir indépendamment du reste.

2.7.3 Avantages de la Méthode.

Cette méthode structurée nous permet de travailler de manière progressive et efficace, garantissant un rendu de qualité tout en maintenant une productivité élevée.



(a) Avant



(b) Après

Figure 12: Comparaison avant/après de la tourelle.

2.7.4 Processus d'Exportation.

Ensuite, vient l'exportation de l'objet vers **Unity**, suivie de l'ajout des textures, de l'assemblage final des pièces et de leur ameublement. Ces objets sont ensuite stockés en tant que préfabriqués et utilisés lors de la génération aléatoire des salles dans le jeu.

2.7.5 Problèmes Rencontrés.

Durant la conception des modèles, nous avons fait face à divers problèmes :

- **Position de base des objets** : La position de base des objets composant une salle importée changeait, la position de base étant définie comme le centre de la salle et non le centre de l'objet lui-même. Cela entraînait une difficulté supplémentaire lors du déplacement des objets.
- **Temps de production d'un objet** : Les objets plus grands ou plus détaillés nécessitent davantage de temps pour leur création.
- **Correspondance avec les besoins réels** : Il était parfois difficile d'ajuster la production des objets en fonction de nos besoins réels de conception.

2.7.6 Objectifs pour la Prochaine Soutenance.

Pour la prochaine soutenance, nous prévoyons de terminer un certain nombre de salles de base et d'objets, ce qui permettra d'améliorer le niveau de diversité graphique des environnements 3D du jeu.

2.7.7 Exemples

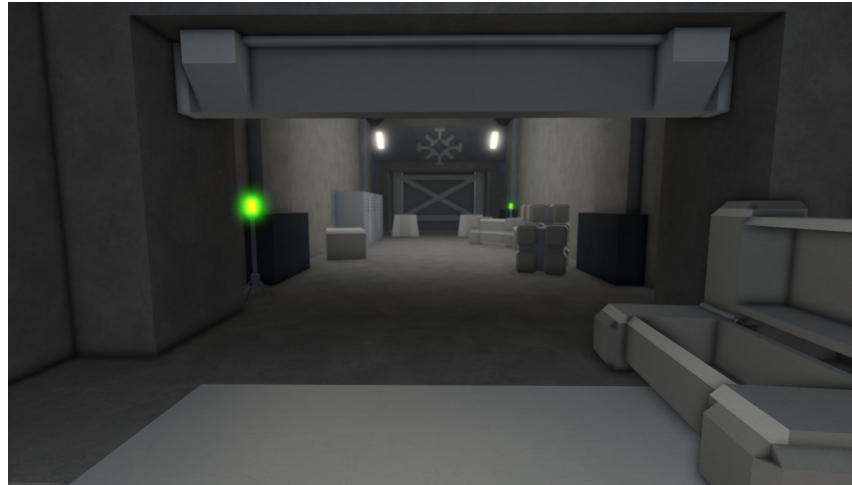


Figure 13: Exemple de couloir”



Figure 14: Salle d'accueil où les joueurs apparaissent en se connectant

3 Conclusion

3.1 Bilan sur l'avancement

Malgré un avancement inférieur à nos attentes initiales, cette première phase du projet nous a permis de poser les bases solides pour la suite du développement. Les défis rencontrés, tels que les difficultés techniques avec le multijoueur ou la réactivité des interfaces graphiques, ont été autant d'opportunités d'apprentissage et de renforcement des compétences de l'équipe.

Nous avons également progressé sur plusieurs aspects cruciaux :

- La mise en place d'un système multijoueur fonctionnel et évolutif grâce à l'intégration de NetCode for GameObjects.
- La conception de plusieurs interfaces utilisateur essentielles, telles que le menu principal et les interfaces des énigmes.
- L'élaboration des premières fonctionnalités liées aux mouvements et interactions des joueurs, posant ainsi les bases des mécaniques de gameplay.
- La structuration du site web du projet, un outil indispensable pour promouvoir le jeu et interagir avec notre future communauté.

Ces avancées reflètent un effort collectif qui, bien que perfectible, témoigne de notre engagement et de notre capacité à surmonter les imprévus.

3.2 Perspectives et futur du projet

Pour la prochaine soutenance, prévue en mars 2025, notre priorité est de proposer une version jouable complète de notre jeu. Ce jalon représente une étape critique, car il nous permettra :

- D'intégrer et de tester l'ensemble des fonctionnalités prévues.
- De collecter des retours précieux à travers une bêta publique, en ciblant particulièrement les aspects de jouabilité, de stabilité et de satisfaction utilisateur.
- D'optimiser les performances du jeu, notamment en multijoueur, et de résoudre les derniers bugs bloquants.

Pour atteindre ces objectifs, nous avons décidé de renforcer notre gestion de projet en adoptant un calendrier précis, avec des échéances définies pour chaque fonctionnalité. Cette méthode de travail nous permettra de mieux répartir les charges et de maintenir une cadence de développement soutenue tout en minimisant les retards.

3.3 Réflexion et apprentissages

Ce premier cycle de développement a mis en lumière l'importance d'une **communication fluide**, d'une **organisation rigoureuse** et d'une **anticipation des contraintes techniques**. Chaque membre de l'équipe a pu développer ses compétences, que ce soit en **gestion de projet**, en **développement logiciel** ou en **conception artistique**. Ces enseignements nous seront indispensables pour réussir les prochaines phases.

Également, cela a permis à tous les membres du projet de se familiariser avec les commandes **git**, ainsi qu'avec l'utilisation des branches dans un projet où le travail de chacun est à la fois **individuellement** et **collectivement** important.

Enfin, travailler sur ce projet nous a tous motivés à continuer de manière plus intense, maintenant que nous connaissons davantage les outils à notre disposition et vers où nous pouvons nous orienter.

En conclusion, bien que cette première phase ait présenté des défis, elle constitue un **socle prometteur** pour la suite. Nous sommes désormais mieux préparés à mener ce projet à son terme et à livrer un jeu captivant et de qualité, fidèle à notre vision initiale.