

---

**RAPPORT DE PROJET 2023**  
**ATIPE**

**Robin R. / Fabien R. / Mehdi T. / Yassin H.**

---

## **Tables des matières**

1. Introduction	1
2. Origine et nature du projet	2
3. Objet de l'étude	3
4. Etat de l'art	
5. Organisation	4
a. Logiciels utilisées	
b. Méthode de travail	
6. Travaux accomplies	6
a. Mehdi	
b. Fabien	
c. Yassin	
d. Robin	
7. Récit de la réalisation	
a. Joies	
b. Peines	
8. Aspect Économique	7
9. Conclusion	

## **Introduction :**

Qui n'a jamais rêvé d'avoir les sujets des partiels en avance. La célèbre école d'informatique epita réputée pour sa difficulté et sa rigueur, a plongé de jeunes SUP dans un sacré pétrin...

Dans ce jeu vous incarnerez un épitéen en grande difficulté scolaire qui doit, s'il veut valider son année, récupérer les précieux sujets des partiels !

Ce malheureux a fait appel à son meilleur copain de classe qui trouve l'idée beaucoup trop périlleuse. Le second joueur incarnera ce personnage indispensable dans le déroulé du jeu. Ce dernier, trop peureux pour se lancer dans une telle épopée, guidera son ami depuis chez lui, grâce à ses nombreuses facultés informatiques. Les deux compères seront liés en vocal durant toute la partie. Et devront résoudre ensemble des énigmes afin de parvenir à trouver ces fameux sujets.

Le vaste Campus situé au 86 Boulevard Vivier Merle, est plongé dans le noir. Notre héros se réveille au rez-de chaussé, il s'est caché dans un placard situé dans le réfectoire. Mission accomplie, le gardien ne l'a pas remarqué pour l'instant. Il est deux heures passées et Epita a fermé ses portes depuis deux heures. Malheureusement vous vous êtes réveillé trop tard...

Votre téléphone qui vous permet de communiquer avec votre ami va bientôt s'éteindre. Il vous reste une heure pour récupérer ce que vous cherchez et fuir. Bonne chance ! ou plutôt bonne nuit...

## **Origine et nature du projet :**

Ce projet est né de l'intention de vouloir produire une référence pour l'EPITA. Tout d'abord, l'idée était de créer un Espace Numérique de Travail (ENT), où les étudiants de toutes promotions confondues auraient accès à leurs notes, leur emploi du temps en fonction de leur classe et de leurs options, avoir sous la main les documents numériques importants ainsi que communiquer avec leurs professeurs, et à terme être disponible aussi bien sur un site que sur une application. Cela aurait permis de réunir les différents dispositifs qui proposent l'EPITA en une seule interface.

Or la dimension multijoueur étant exigée, nous n'avons pas baissé les bras, toujours dans l'optique d'être une référence pour l'EPITA. C'est de là qu'en a découlé la volonté de modéliser le campus d'EPITA Lyon, et pour combler cette appétence aussi ambitieuse que profitable, quoi de mieux que d'en faire un jeu vidéo ? A cela s'ajoute l'exaltation pour modéliser nos professeurs, le vigile avec qui nous avons sympathisé etc... Il s'agira donc d'un jeu constitué de deux joueurs agissant en coopération.

## **Objet de l'étude :**

Ce projet, qui se caractérise comme étant un concept vidéo ludique, a pour but d'accentuer la réflexion du joueur lors de ses utilisations. En effet, l'idée même de la création de notre projet se base principalement sur la résolution d'énigme, la patience mais également sur la persévérance. Il s'agit ici d'un atout capital que le joueur peut ou non maîtriser car il s'agit ici d'apprendre à ce dernier de les développer et de les améliorer afin de pouvoir poursuivre pleinement la continuité de la partie. Ce projet nous apprend également, à l'échelle individuelle, à connaître nos capacités mais également à ne pas se décourager face aux différents obstacles et à persister afin de résoudre les multiples contrebemps durant la conception. Mais aussi, ce projet nous apporte à l'échelle collective un moyen de nous surpasser et de travailler en équipe en laissant nos différends de côté et apprendre à connaître nos partenaires pour travailler ce qui joue en faveur du développement des relations sociales, mais développe également nos capacités à se fixer et essayer d'atteindre des objectifs ambitieux.

## **État de l'art :**

### a. Origine

Le tout premier jeu multijoueur comportant des énigmes et disponible directement sur navigateur est Quests Islands. Mais le principal jeu qui nous a donné envie d'implémenter un système d'énigmes à résoudre pour progresser dans le jeu fut Tiny Room, un jeu mobile que nous connaissions déjà du même type qu'un jeu accessible sur console et sur PC : The Rooms. Il est également semblable à Mist, de par l'aspect découverte, se retrouver dans un endroit inconnu, et parvenir à se débrouiller par ses propres moyens de fil en aiguille à la manière d'un Escape Game. Il est tout de même bon de noter qu'ATIPE se démarque de ces derniers de par sa jouabilité à plusieurs et que ceux-ci ne sont que des bases pour mieux appréhender le projet. Les points forts de ces jeux ne sont pas tant dans les graphismes emphatiques (exceptés pour les derniers jeux Mist sortis) mais dans le scénario et la mise en relations des différents éléments qui rendent les jeux complets. A cela s'ajoute le tactile pour Tiny Room et The Rooms ainsi que l'ambiance angoissante posée dans Mist.

### b. Fonctionnement

Deux joueurs jouent en coopération. Ils se trouvent dans deux endroits différents. La communication est la clé de la réussite c'est pourquoi il sera nécessaire de créer un bon moyen de communication via deux joueurs jouant sur des machines différentes.

Des petites énigmes s'enchaînent, à la fois physiques et mentales dans une ambiance angoissante pour le joueur s'introduisant dans epita. Le joueur qui l'aidera sera dans un appartement étudiant, il guidera son collègue en réalisant des énigmes lui aussi.

Le but du joueur sera de gravir les étages, et d'arriver au 6ème étage. Nous allons revisiter les locaux d'EPITA de sorte à ce que le jeu soit ludique.

L'IA sera le gardien d'Epita, un indice sonore permettra au joueur de savoir si le gardien est à proximité ou non. Ce dernier aura une hitBox constamment devant lui dès lors que le joueur touche cette hitBox le jeu se termine et le joueur perd.

Du côté du personnage un inventaire épuré sera réalisé avec peu d'objets. Un système de saut sera disponible avec la possibilité de se mettre accroupi. Pour des raisons évidentes, aucun système de combat ne sera implémenté.

Enfin, un système de caméra sera implémenté ce qui permettra au joueur qui aide l'autre depuis chez lui de visualiser certaines salles.

## Organisation :

Notre cahier des charges fut légèrement modifié.

Voici sa version finale.

SOUTENANCE		R.R	F.R	M.T	Y.H	Total
1ère	Programmation du Jeu Game design Squelette du Jeu Création du terrain Déroulé des énigmes Animations 3D	5% 15% 15% 10% 10% 25% 5%	5% 5% 10% 25% 15% 25% 15%	5% 15% 15% 25% 25% 25% 20%	5% 5% 10% 10% 10% 25% 20%	20% 25% 35% 35% 100% 20%
2ème	Programmation du Jeu Game Design Squelette du Jeu Création du terrain Implémentation enigmes Conception de L'I.A. Multijoueur Audio Communication Animations 3D	10% 15% 30% 15% 20% 15% 50% 10%	10% 10% 10% 15% 35% 10% 25% 25%		10% 10% 10% 15% 15% 15% 10%	50% 60% 85% 100% 50% 50% 50% 60%
3ème	Programmation du Jeu Game design Squelette du Jeu Implémentation enigmes Création du terrain Conception de L'I.A. Multijoueur Audio Communication Cinématique Déroulé enigmes Animations 3D	25% 20% 15% 50% 50% 50% 25%	10% 10% 50% 15% 35%		15% 10% 25% 25% 50%	100% 100% 100% 100% 100% 100% 100% 100% 100% 100%

Voici l'ancien cahier des charges :

SOUTENANCE		R.R	F.R	M.T	Y.H	Total
1ère	Programmation du Jeu <b>Game design</b> Squelette du Jeu <b>Création du terrain</b> Déroulé des énigmes Animations 3D Recherche sur I.A. Recherche Multijoueur	5% <b>5%</b> <b>50%</b> 25% 5% <b>50%</b>	5% <b>10%</b> <b>20%</b> 25% 5% <b>50%</b>	5% <b>10%</b> <b>10%</b> 25% <b>5%</b> <b>50%</b>	5% <b>10%</b> <b>5%</b> 25% 5% <b>50%</b>	20% <b>35%</b> <b>85%</b> 100% 20% <b>100%</b>
2ème	Programmation du Jeu <b>Game Design</b> <b>Squelette du Jeu</b> Création du terrain Implémentation enigmes <b>Conception de L'I.A.</b> <b>Multijoueur</b> Audio Communication Animations 3D	<b>10%</b> <b>40%</b> 15% <b>50%</b> 10% <b>30%</b>	10% 35% <b>50%</b> 15% <b>30%</b>		10% 30% <b>25%</b> 15% 10%	50% <b>85%</b> 100% 50% 50% <b>100%</b> 50% <b>100%</b>
3ème	Programmation du Jeu <b>Game design</b> Squelette du Jeu Implémentation enigmes Création du terrain <b>Conception de L'I.A.</b> <b>Multijoueur</b> Audio Communication Cinématique Déroulé enigmes <b>Animations 3D</b>	15% <b>10%</b> <b>50%</b>	10% 50% 15% 20% 80%	10% 5% <b>25%</b> <b>50%</b>	15% <b>100%</b> 100% 100% 100% 100% 100% <b>100%</b>	100% 100% 100% 100% 100% 100% 100% <b>100%</b>

- **Eléments ajoutés**
- **Eléments supprimé**
- **Eléments modifiés**

Nous avons pu affiner notre découpage du projet avec une meilleure répartition des tâches. Deux personnes travaillent sur une tâche plutôt que tout le monde travaille sur la même. Cela est beaucoup plus efficace.

## **Répartition sur trois soutenances :**

### **Première : 10 mars 2023**

- Modélisation du terrain
- Modélisation des personnages
- Création des animations
- Création des squelettes joueurs
- Familiarisation avec Unity

### **Deuxième : 21 avril 2023**

- Début réalisation de l'IA
- Implémentation du multijoueur
- ajout de cinématique
- Avancé Squelettes joueurs
- Avancé modélisation personnages
- Audio Communication entre joueur
- 

### **Finale : du 29 mai 2023 au 9 juin 2023**

- Fin de l'IA et implémentation dans le jeu
- Multijoueur fonctionnel
- Début cinématique
- Squelettes joueurs terminée
- Modélisation des Personnage terminée
- Possibilité de communiquer / implementation son
- Écran menu / pause / (settings si jeux finis)

## Script

### Sommaire (chronologique)

Extérieur	RDC	Etage 1	Etage 2	Etage 3	Etage 4
Introduction dans Epita					

Personnage 1: personnage qui veut récupérer les informations des partiels.

Personnage 2: personnage qui est chargé de guider son coéquipier dans Epita.

Les deux personnages sont en communication à partir du moment où le vocal s'active et ils le resteront toute la partie. (Vocal) Lorsque le joueur à Epita se fera toucher par le gardien, il reviendra à l'entrée mais NE PERDRA PAS SA PROGRESSION. (ceci nous évite les conflits de sauvegarde trop complexes. )

### Extérieur

Personnage n°1 dans la rue. L'atmosphère est angoissante, avec du brouillard et des lampadaires émettant une faible lumière pendant la nuit. Le personnage explore et se dirige vers l'entrée d'Epita. Il trouve un papier qui dit : "Le gardien est présent ici toute la nuit, il semblerait qu'il détienne un double des clés du campus dans sa voiture". Il doit donc trouver la voiture du gardien. Un message apparaît en bas de l'écran : "Hmmm... la voiture est fermée, mais il me semble avoir vu une caisse à outils traîner devant. Je pourrais jeter un coup d'œil". Le personnage récupère un pied-de-biche et s'en sert pour ouvrir la voiture et récupérer les clés. Il peut alors pénétrer dans Epita.

Le personnage n°2 se trouve dans sa chambre. Il reçoit un message de son coéquipier. En explorant la pièce, il découvre son ordinateur. Il doit le déverrouiller. Lorsque le personnage accède à son ordinateur, il est transporté vers une nouvelle scène, représentant un écran d'ordinateur. Le joueur devra tout d'abord déverrouiller l'ordinateur pour pouvoir y accéder. Une fois déverrouillé, il aura accès aux caméras de surveillance d'Epita. Cependant, le joueur a fait la fête la veille et il a du mal à se souvenir de son mot de passe. Pour récupérer ce mot de passe, il devra chercher des indices dans sa chambre. Parmi ces indices, il y a un rébus à résoudre :



solution : 1587

## Rez De Chaussée

Le joueur numéro 1 arrive à Epita. Le gardien dort, donc l'IA n'est pas activée. La porte est entrouverte et le gardien dort. Le personnage se trouve dans l'obscurité. Il trouve une lampe torche pour mieux voir. Son objectif est de se rendre au réfectoire. Il doit trouver un disjoncteur. Lorsqu'il entre, un message s'affiche : "Il fait sombre ici. Je crois avoir déjà vu un disjoncteur près de la machine à café." Il trouve le disjoncteur, qui a un modèle spécifique. Le joueur devra transmettre le modèle à son coéquipier. Son ami aura accès au mode d'emploi du disjoncteur, qui contiendra un code couleur à entrer. Cela permettra d'allumer les lumières.

Le personnage n°2 a accès au PC. Il devra fouiller dans les fichiers afin de trouver le mode d'emploi du bon modèle de disjoncteur. Il renseignera le code à son coéquipier.

## 1er étage

Le personnage n°1 devra suivre les indications de son coéquipier. Au premier étage, il trouvera une affiche contenant des informations sur les changements à Epita :

"""

### GRANDS CHANGEMENTS À EPITA

- Rénovation des toilettes
- Atelier de création de jeu de palet
- Ouverture d'un nouveau club de Polo
- Nouveaux casiers situés au rez de chaussée
- Soirée au Perroquet Bourré le 31 février

"""

Cette affiche lui indique de trouver les casiers. Il obtient le code grâce à son coéquipier. Le joueur détient une clé USB. Avec l'aide de son coéquipier, le joueur devra trouver les serveurs d'Epita. Les serveurs se trouvent dans une pièce secrète, située dans les escaliers reliant le rez-de-chaussée et le premier étage. Le logo IONIS est une porte d'accès à cette pièce. Pour ouvrir la porte, le joueur passera devant la machine à café où il verra affiché : "Tient, la machine à café a l'air fonctionnelle, pour une fois..." Un bouton sur cette même machine permettra d'ouvrir la porte. Il insérera ensuite la clé USB à l'intérieur, ce qui débloque l'accès aux caméras du premier étage pour le joueur numéro 2. Cependant, cela activerait également l'arrivée du gardien.

Le joueur n°2 aura accès à une application de caméra. Au début, l'application sera bloquée. Il devra fouiller dans ses fichiers et trouver un dossier intitulé "Projet Hacking Caméra". Ce dossier contiendra un fichier README avec l'inscription suivante : "Code casier : \*\*\*\*"

Dans ce même dossier, il y aura un fichier indiquant que la clé USB devra être insérée dans les serveurs d'Epita.

## 2eme Etage

Le joueur n°1 devra se rendre dans les bureaux des professeurs pour trouver une clé permettant d'accéder au prochain étage. Les portes seront déverrouillées une fois que les caméras seront activées. En arrivant devant les bureaux, le joueur verra affiché : "Il me semble que je suis devant les bureaux des professeurs. J'ai entendu dire que M. Saber travaillait tard, donc je dois être vigilant." Le joueur aura donc deux choix pour ouvrir une porte. Il pourra ouvrir le bureau intitulé "Professeur", où il y aura un professeur en train de travailler. Si le joueur entre dans cette pièce, il perdra la partie. Le bon choix sera d'entrer dans la salle de la secrétaire. À l'intérieur, le joueur sera guidé par son coéquipier pour fouiller le bureau de la secrétaire.

Le joueur numéro 2 aura accès à un dossier d'archives contenant d'anciennes vidéos provenant des caméras. Parmi ces vidéos, l'une d'entre elles indiquera l'emplacement de la clé. Plus précisément, il s'agira d'une vidéo montrant la secrétaire qui place sa clé dans un endroit spécifique.

## 4eme Etage

Le joueur n°1 devra tenter d'allumer le projecteur en se dirigeant vers les rangs de l'amphithéâtre pour trouver un ordinateur. Il découvre un sac oublié par un élève, dans lequel se trouve l'ordinateur nécessaire.

Pour le joueur numéro 2, le PC affichera un problème de réseau, ce qui obligera le joueur à quitter son PC et à résoudre le problème. Il sortit de sa chambre pour trouver une solution au problème de réseau.

## Fin

Le joueur retrouve l'usage de son PC et découvre une nouvelle application permettant d'ouvrir une pièce secrète à Epita.

Le joueur récupère les partiels, débloque la sortie et peut s'échapper tout en fuyant le vigile.

## **Travaux accomplies:**

### a. Mehdi

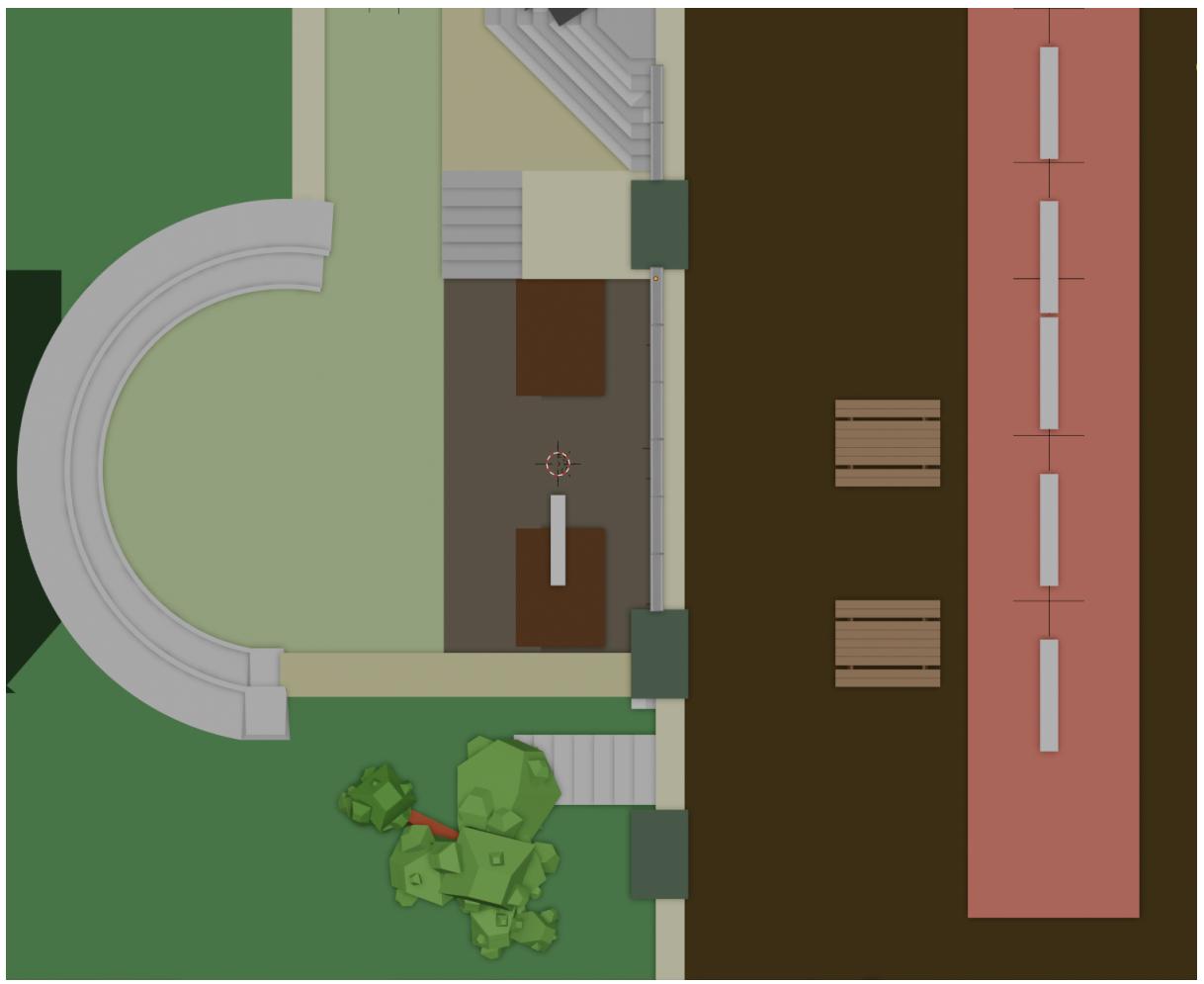
J'ai pour ma part participé à la modélisation de la chambre afin de lui donner l'apparence qui est actuellement présente dans le jeu. Cette chambre sera le lieu d'appréhension du joueur qui mènera l'opération afin de pouvoir gagner la partie

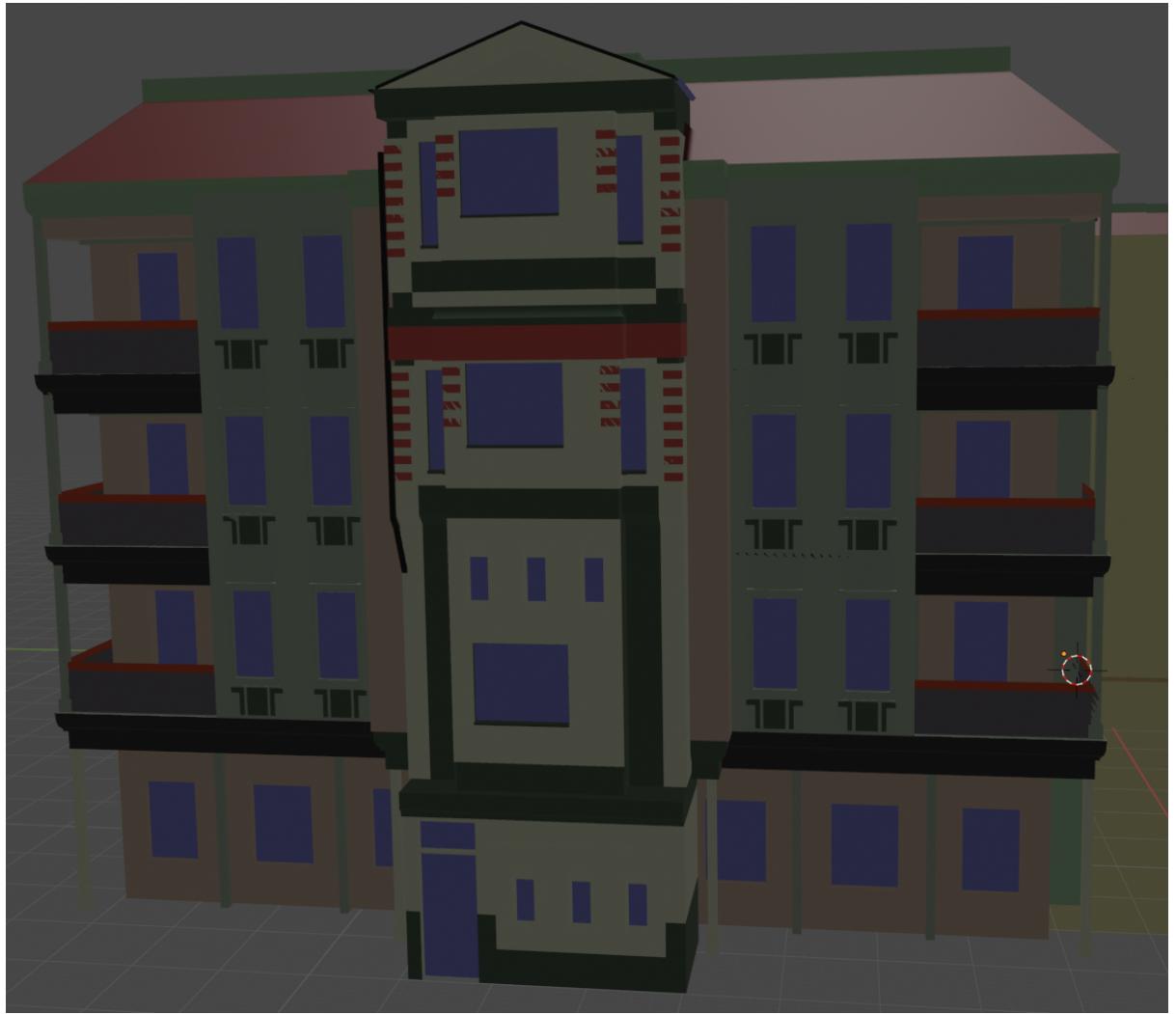
Nous avons décidé d'ajouter un chat vocal pour permettre une communication fluide entre les deux joueurs. Pour cela, nous devions trouver un package compatible avec Mirror (notre game networking).

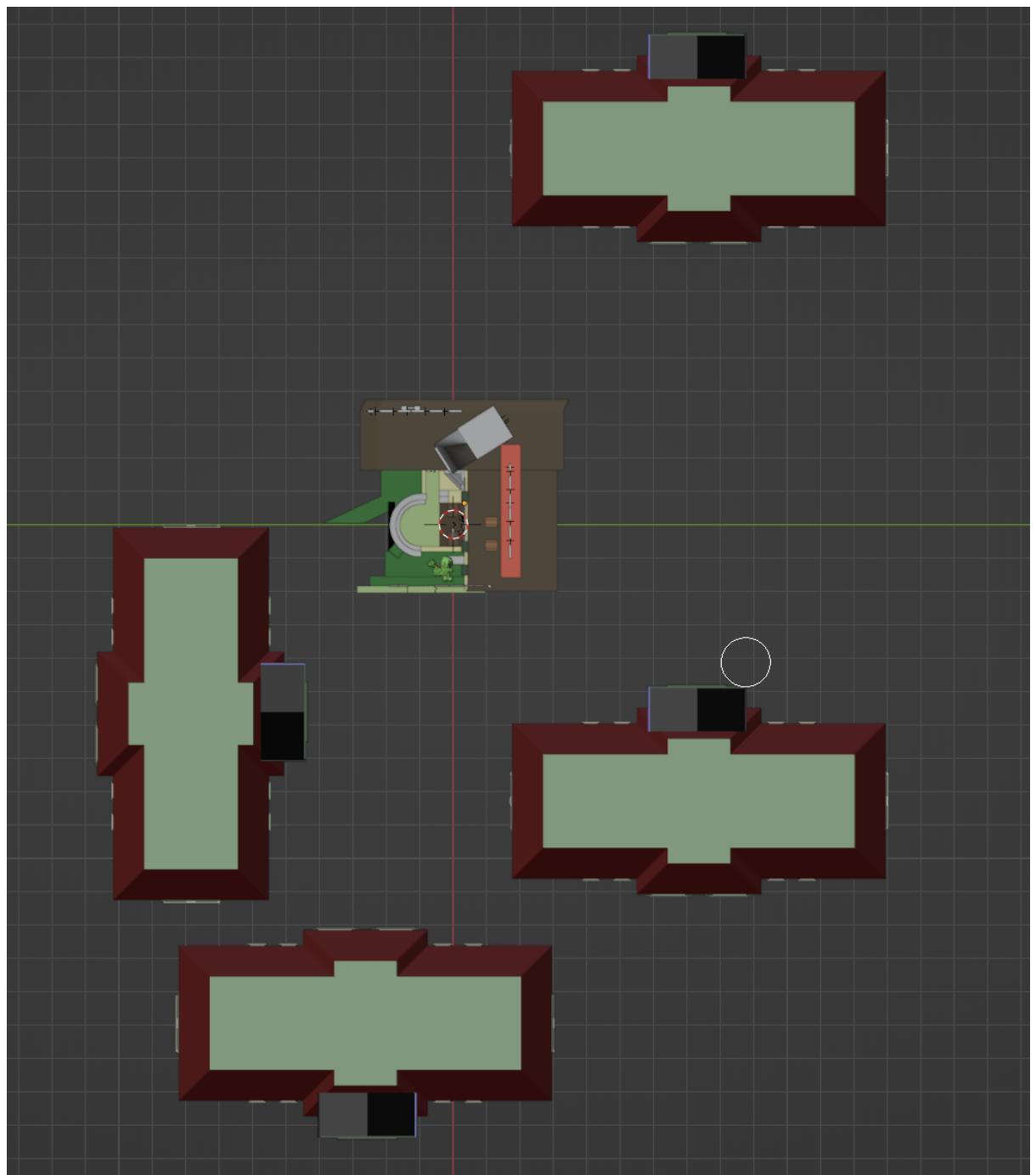
Nous avons donc décidé d'utiliser l'asset Dissonance Voice Chat. Nous ferons une démonstration pendant la soutenance

### b. Fabien

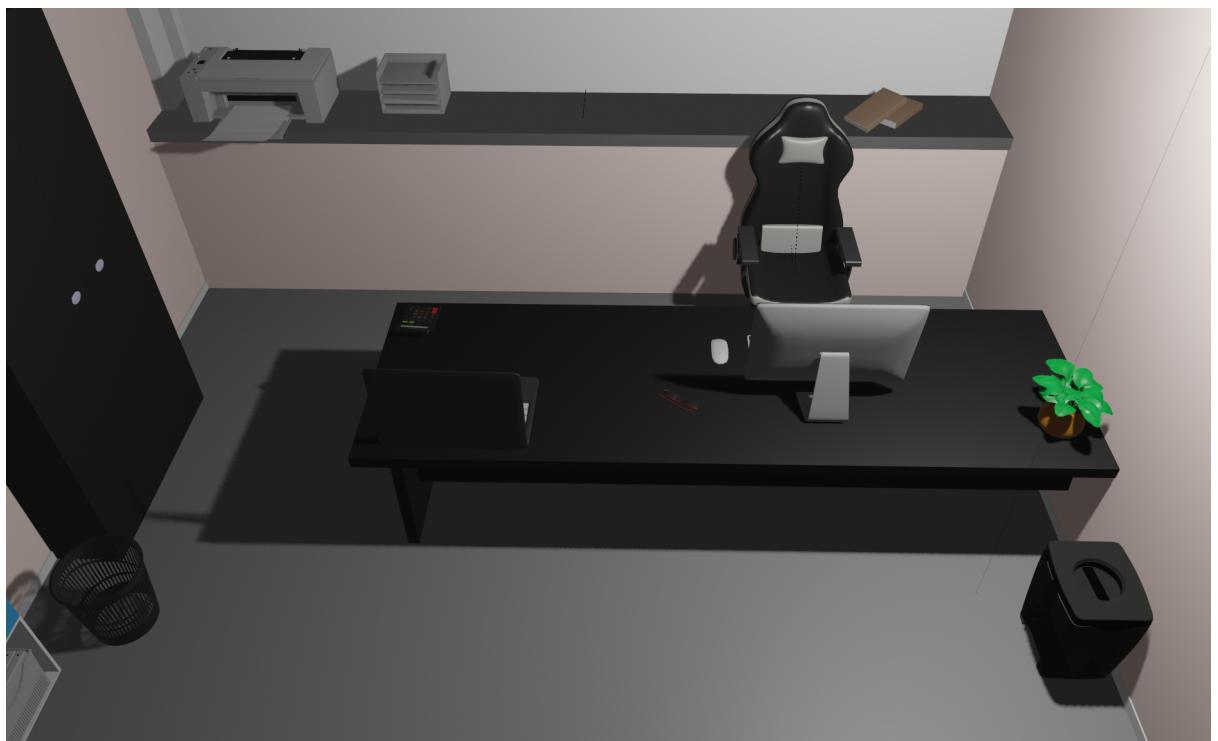
Je me suis occupé principalement de la modélisation et de l'animation



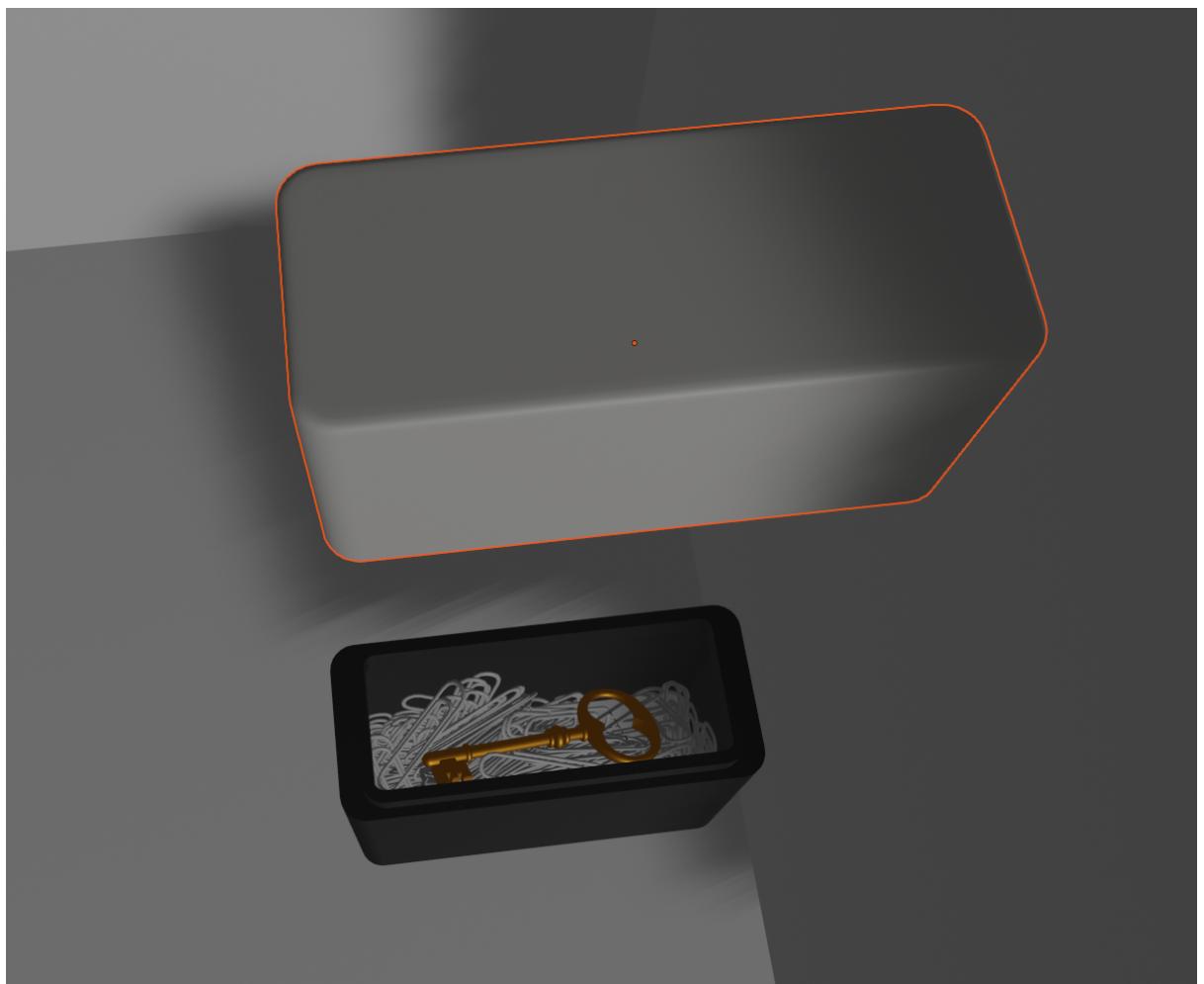












### c. Yassin

#### Intelligence Artificielle

L'un des éléments les plus importants de notre jeu est le gardien, c'est lui que le joueur 1 devra éviter afin de remporter la partie. Pour créer le gardien, nous avons créé une intelligence artificielle, avec 2 états fondamentaux, l'état patrolling, qui correspond à l'état de base du gardien et l'état chase, qui correspond à l'état lorsque le gardien a détecté le joueur. A la première soutenance, nous avons présenté un script très basique de l'IA, qui prenait également plus de ressources car n'utilisait pas le module NavMesh (qui permet de définir si des endroits sont exploitables pour marcher). A la deuxième soutenance, nous avons beaucoup amélioré le gardien, mais sans l'implémenter.

```
private void Awake()
{
    player = GameObject.Find("players").transform;
    agent = GetComponent<NavMeshAgent>();
}

private void Update()
{
    playerInSightRange = Physics.CheckSphere(transform.position, sightRange, whatIsPlayer);

    if (!playerInSightRange )
    {
        Patroling();
    } else
    {
        Chase();
    }

    animator.SetFloat("speed", agent.desiredVelocity.sqrMagnitude);
}

private void Patroling()
{
    if (!walkPointSet) SearchWalkPoint();

    if (walkPointSet)
    {
        agent.SetDestination(walkPoint);
    }

    Vector3 distanceToWalkPoint = transform.position - walkPoint;
    if (distanceToWalkPoint.magnitude < 1f) walkPointSet = false;
}

private void SearchWalkPoint()
{
    Vector3 randomDirection = Random.insideUnitSphere * 35;
    randomDirection += transform.position;
    NavMeshHit hit;
    NavMesh.SamplePosition(randomDirection, out hit, 35, 1);
    walkPoint = hit.position;
    walkPointSet = true;
}

private void Chase()
{
    agent.SetDestination(player.position);
}
```

Enfin, pour cette dernière soutenance, nous avons effectué quelques modifications sur le script et surtout nous avons implémenté l'IA dans le jeu.

```
void Start()
{
    player = GameObject.FindGameObjectWithTag("Player").transform;
    navMeshAgent = GetComponent<NavMeshAgent>();
}

void Update()
{
    if (CanSeePlayer())
    {
        FollowPlayer();
    }
    else
    {
        Wander();
    }
    animator.SetFloat("speed", navMeshAgent.desiredVelocity.sqrMagnitude);
}

bool CanSeePlayer()
{
    if (Vector3.Distance(transform.position, player.position) <= sightRange)
    {
        RaycastHit hit;
        if (Physics.Raycast(transform.position, player.position - transform.position, out hit, sightRange))
        {
            if (hit.collider.CompareTag("Player"))
            {
                return true;
            }
        }
    }
    return false;
}
```

```
void FollowPlayer()
{
    navMeshAgent.SetDestination(player.position);
    navMeshAgent.speed = runSpeed;
    if (Vector3.Distance(transform.position, player.position) <= stoppingDistance)
    {
        navMeshAgent.isStopped = true;
    }
    else
    {
        navMeshAgent.isStopped = false;
    }
}

void Wander()
{
    if (!walkPointSet)
    {
        float randomZ = Random.Range(-walkRadius, walkRadius);
        float randomX = Random.Range(-walkRadius, walkRadius);
        walkPoint = new Vector3(transform.position.x + randomX, transform.position.y, transform.position.z + randomZ);

        if (!NavMesh.SamplePosition(walkPoint, out NavMeshHit hit, 1f, NavMesh.AllAreas))
        {
            return;
        }

        walkPointSet = true;
    }

    navMeshAgent.SetDestination(walkPoint);
    navMeshAgent.speed = walkSpeed;

    if (Vector3.Distance(transform.position, walkPoint) < 1f)
    {
        walkPointSet = false;
    }
}
```

(Nous avons changé le nom des fonctions pour plus de clarté)

Important : A l'heure actuelle, le gardien n'est pas modélisé et nous utilisons des assets gratuits trouvés sur internet. Nous mettons tout en œuvre pour avoir terminé d'ici le 1er juin, mais actuellement la modélisation du gardien est dernière dans notre liste de priorité.

### **Chat vocal**

Nous avons décidé d'ajouter un chat vocal pour permettre une communication fluide entre les deux joueurs. Pour cela, nous devions trouver un package compatible avec Mirror (notre game networking).

Nous avons donc décidé d'utiliser l'asset Dissonance Voice Chat. Nous ferons une démonstration pendant la soutenance

### **Site Web**

atipe.fr a été mis en ligne lors de la deuxième soutenance, il avait un design basique et manquait de plusieurs fonctionnalités. Le design a été revisité et le site est désormais meilleur que ce soit esthétiquement ou techniquement.



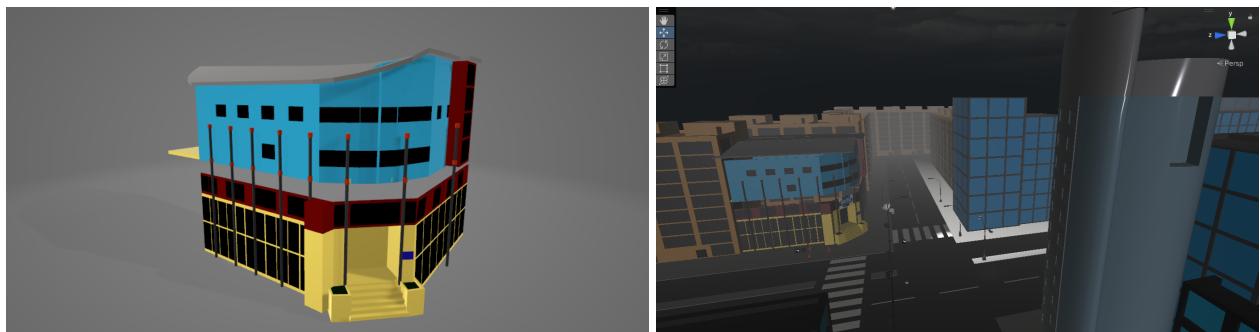
## d. Robin

### Modélisation

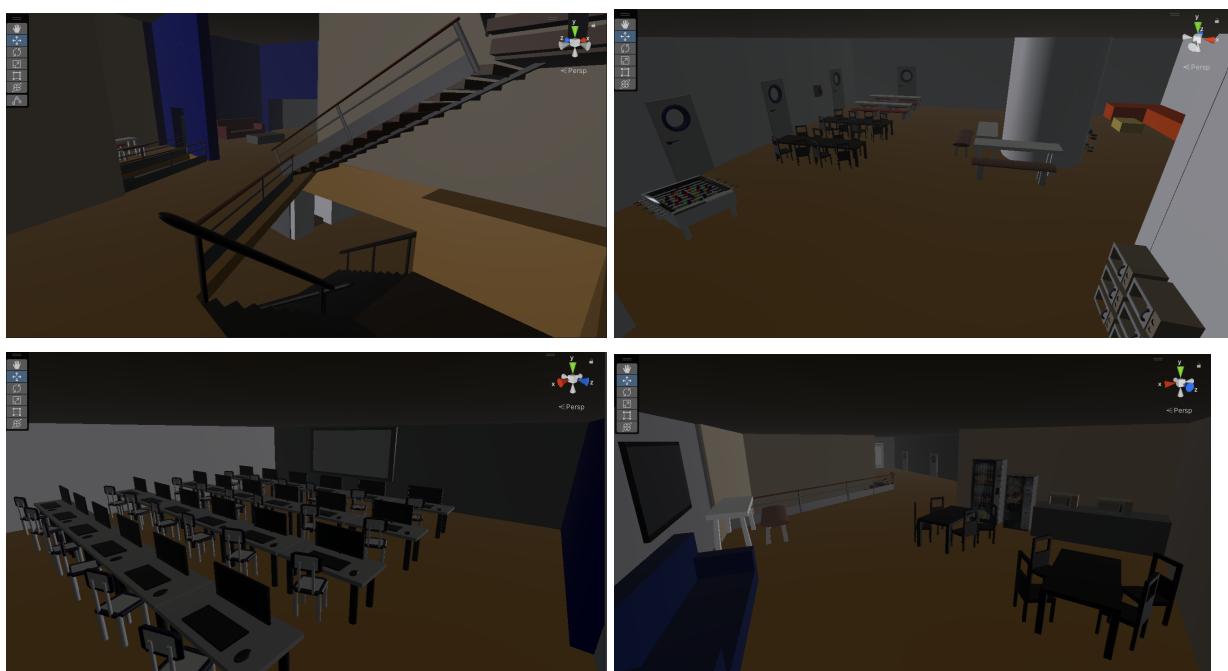
Pour modéliser le jeu j'ai tout réalisé sur Blender. Nous avons créé la plupart des objets. Néanmoins nous avons fait appel à des sites gratuits qui proposent des modèles 3D. Par exemple, TurboSquid, Open 3D Model, Free 3D ont servi pour les objets les plus complexes, comme les plantes par exemple.

Ici voici une liste qui ont été réalisé par moi-même :

- Façade Epita / Extérieur Epita



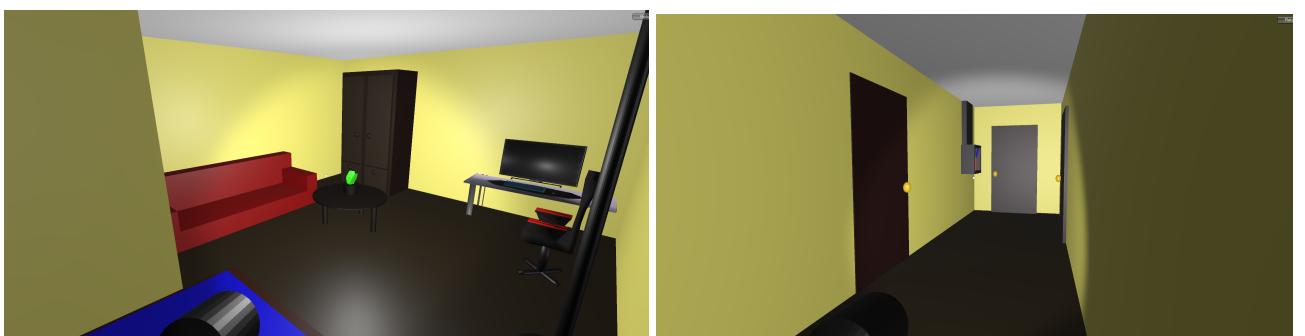
- Intérieur Epita



- Amphithéâtre



- Chambre Joueur 2



- Objets divers



Voici la liste complète des objets que j'ai modélisé sur Blender pour la conception du jeu :

- Armoire
- Lit
- Banc Epita
- Barrière
- Boîte à outils
- Canapé
- Casier
- Chaise cours
- Chaise amphi
- Clé à molette
- Escaliers (rez de chaussée vers premier + premier vers second étage)
- Façade Epita
- Clé
- Lampadaire
- Lampe de poche
- Logo Ionis
- Machine à café
- Pied de biche
- Porte Epita
- Porte chambre
- Poubelle
- Poste salle machine
- Server
- Tableau
- Télé
- Visse
- Visseuse

## Programmation Joueurs

J'ai commencé la programmation en créant le squelette du joueur. Voici comment j'ai divisé la programmation du personnage :

### 1. Contrôleur du joueur (Player Controller) :

Dans cette partie, je gère les touches pressées par le joueur pour le déplacer en utilisant les commandes "GetAxisRaw". J'ai également utilisé la structure Vector3 déjà implémentée dans Unity. Cette structure est utilisée pour transmettre des positions et des directions en 3D. Elle contient également des fonctions pour effectuer des opérations vectorielles courantes. Dans cette classe, je gère également la rotation du joueur en prenant en compte les mouvements de la souris. Ici, nous remarquons que le joueur peut se déplacer uniquement lorsque le booléen "CanMoveHead" est vrai. L'utilisation de ce booléen est extrêmement efficace lors de la mise en pause du jeu ou lorsque le joueur utilise l'ordinateur pour le figer sur place.

```
private void Update()
{
    if(CanMoveHead)
    {
        // Calculer la vitesse du mouvement de notre joueur

        //déplacement vers la gauche ou vers la droite
        float xMov = Input.GetAxisRaw("Horizontal");

        //déplacement vers l'avant ou vers l'arrière
        float zMov = Input.GetAxisRaw("Vertical");

        //Dans Unity dans ProjetSettings Possibiliter de changer les Input (azerty/qwerty)

        Vector3 moveHorizontal = transform.right * xMov; //précise l'axe de déplacement
        Vector3 moveVertical = transform.forward * zMov;

        Vector3 velocity = (moveHorizontal + moveVertical).normalized * speed; //on calcule la vitesse du perso

        motor.Move(velocity);

        //Rotation de la camera qui suit la souris

        // On calcule la rotation du Joueur en un Vector3
        float yRot = Input.GetAxis("Mouse X"); // Pourquoi yRot avec Mouse X => axe y modifier mouvement souris x
        Vector3 rotation = new Vector3(0, yRot, 0) * mouseSensitivity;

        motor.Rotate(rotation); //method qui applique la rotation

        // Rotation de la camera car sinon le joueur se penche de haut en bas
        float xRot = Input.GetAxis("Mouse Y");
        float cameraRotationX = xRot * mouseSensitivity;
```

## 2. Moteur du joueur (Player Motor) :

Ce script gère le mouvement du joueur. Il est lié au Contrôleur du joueur (Player Controller), qui s'occupe, comme son nom l'indique, des contrôles du joueur. J'utilise le composant Rigidbody de notre personnage pour lui appliquer les changements de position. La méthode “PerformMovement” est chargée d'effectuer le mouvement.

```
public void Rotate(Vector3 _rotation)
{
    rotation = _rotation;
}

1 référence
public void RotateCamera(float _CameraRotationX)
{
    cameraRotationX = _CameraRotationX;
}

@ Message Unity | 0 références
private void FixedUpdate() //delai a fixer alors que update toute les frame.
{
    PerformMovement();
    PerformRotation();
}

1 référence
private void PerformMovement()
{
    if (velocity != Vector3.zero) // Vector3.zero Le joueur ne fait rien
    {
        // on move le joueur a la position + sa vitesse * depl au fil du temps
        rb.MovePosition(rb.position + velocity * Time.fixedDeltaTime);
    }
}

1 référence
private void PerformRotation()
{
    //on calcule la rotation de la caméra
    rb.MoveRotation(rb.rotation * Quaternion.Euler(rotation)); //quaternion en argument qui gère les rotations * cast rotation
    currentCameraRotationX -= cameraRotationX; //si + bas / haut inversé
    currentCameraRotationX = Mathf.Clamp(currentCameraRotationX, -cameraRotationLimit, cameraRotationLimit); // la rotation de la caméra sera entre -85 et +85

    // on applique la rotation avec le blocage de la caméra
    cam.transform.localEulerAngles = new Vector3(currentCameraRotationX, 0f, 0f);
}
```

### 3. Interaction du joueur (Player Interact) :

Ce script gère l'interaction du joueur avec les objets. J'utilise un Raycast, qui est une ligne invisible constamment présente dans la caméra du joueur. Ainsi, lorsque le joueur aura un objet au centre de son champ de vision, il sera capable d'interagir avec celui-ci. La méthode "DoPickup" vient seulement ajouter à l'inventaire, placé au préalable dans le personnage, l'objet sélectionné et vient le détruire de la map.

```
//Object Inventaire
if (hit.transform.CompareTag("Item"))
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        playerPickupController.DoPickup(hit.transform.gameObject.GetComponent<Item>());
    }
}

④ Script Unity (référence de ressource) | Référence
public class PickupController : MonoBehaviour
{
    [SerializeField]
    private Inventory inventory;

    1 référence
    public void DoPickup(Item item)
    {
        //ajouter l'objet passé à l'inventaire du joueur
        inventory.AddItem(item.itemData);

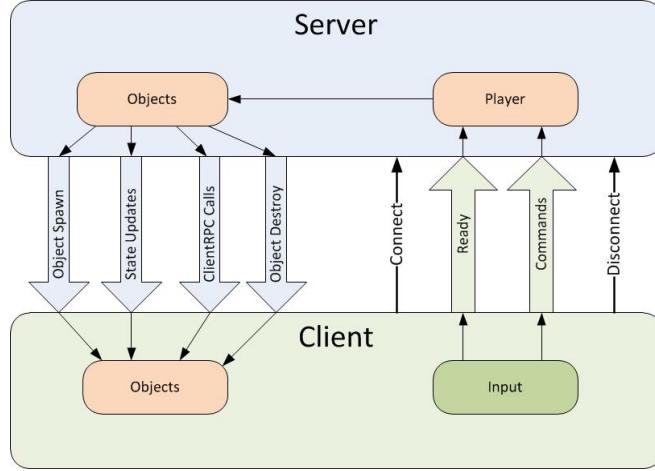
        // détruire objet
        Destroy(item.gameObject);

        // Bloquer le déplacement du joueur pendant qu'on ramasse un objet
    }
}
```

## Multijoueur

Pour synchroniser nos joueurs nous allons utiliser l'asset gratuit proposé par Unity, Mirror. Mirror est un outil efficace afin de générer la mise en réseau du projet. J'utilise deux composants essentiels de Mirror: le "NetworkRoomManager" (permet de gérer le réseau et les scènes) et le "NetworkHUD" (permet de gérer les personnages accueillis sur le réseau). Mirror utilise un système de communication entre chaque objet qui interagit dans le jeu ainsi nous devons ajouter un "NetworkIdentity" à chaque objet utile de notre jeu. (ex : joueur / item / etc ...)

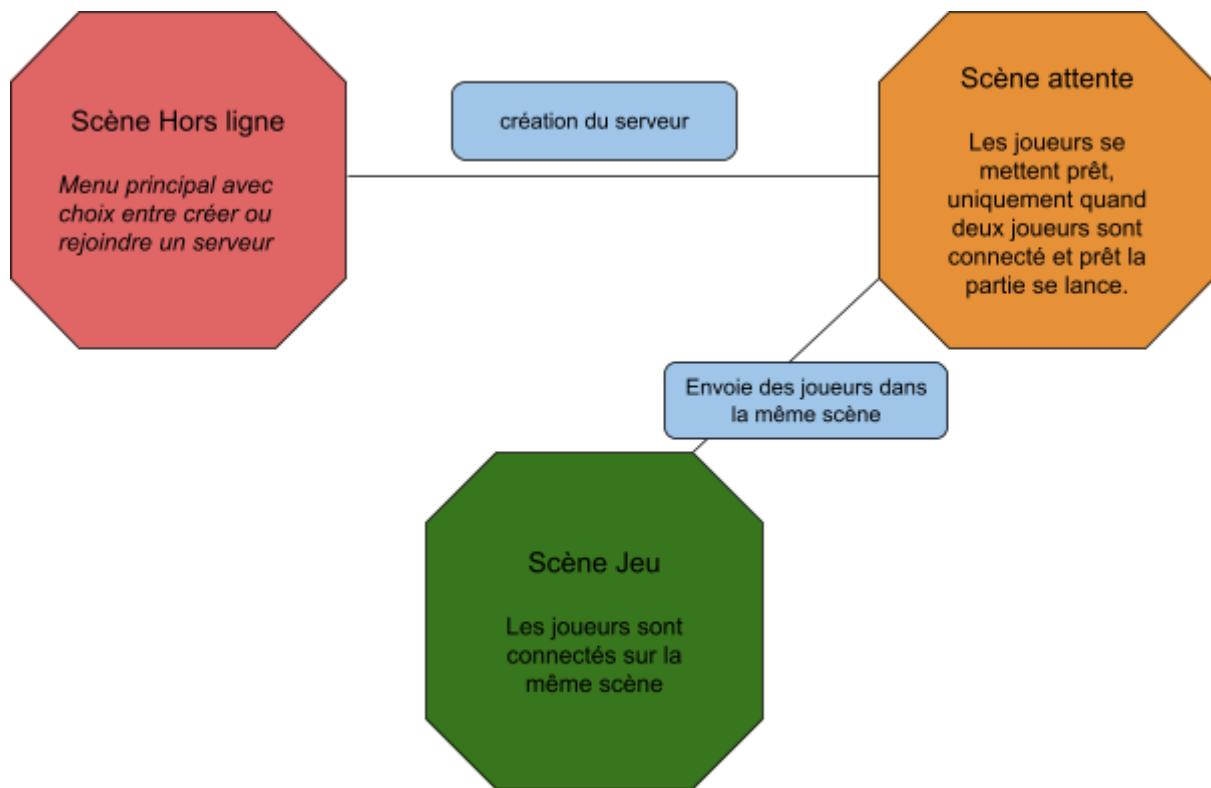
Transmission des informations avec Mirror



Un problème que nous avons rencontré lors de la réalisation du multijoueur est que les joueurs sont connectés les uns avec les autres et dès lors qu'un personnage bouge l'autre aussi. Ainsi nous avons réalisé un script nommé "PlayerSetup", qui sera implémenté sur chacun des joueurs. Le script précisera que les actions réalisées seront faites uniquement sur le joueur sélectionné. En d'autres termes nous préciserons les composants à désactiver pour le joueur.

Du côté de la synchronisation, nous avons choisi d'optimiser les packages envoyés au serveur en les limitant à 100 millisecondes entre chaque envoi. En effet, si nous baissions au maximum ce temps, les serveurs seraient surchargés et cela ferait l'effet inverse les packages seraient triés en file d'attente et le jeu serait saccadé.

Pour une meilleure visualisation de comment la connexion est établie, voici un schéma décrivant la connexion de deux joueurs sur le jeu:



Pour la mise en place du menu principal, j'ai décidé d'utiliser les méthodes déjà pré faites par Mirror. Tout d'abord, j'ai créé un canevas afin de pouvoir adapter les éléments d'interface utilisateur (UI) à chaque résolution d'écran. J'ai utilisé les éléments d'UI suivants : bouton, image et texte (package Text Mesh Pro). J'ai assigné à chaque bouton une méthode spécifique. Par exemple, j'ai assigné la méthode "StartServer()" au bouton "Créer un serveur".

Cependant, j'ai rencontré quelques problèmes :

- Les "RoomPlayers" : Ce sont des objets qui permettent de synchroniser les personnages en multijoueur. Ils se multiplient au fur et à mesure que les joueurs rejoignent le serveur. Pour mettre en place un système de prêt/pas prêt, il était nécessaire de récupérer mon RoomPlayer spécifique et seulement le mien. Pour cela, j'ai limité le nombre de joueurs à 2 et ajouté un tag à mon personnage.
- Les éléments d'UI : Afin de rendre mon jeu plus esthétique, j'ai dû modifier les éléments d'UI, ce qui n'a pas été facile. Dans Unity, nous utilisons un système de Sprite (qui est un format nécessaire pour ajouter des textures aux boutons). J'ai eu du mal à comprendre comment créer ma propre interface utilisateur, alors j'ai utilisé un package gratuit disponible sur Unity Hub.



J'ai découvert la notion de shader lors de l'implémentation du menu pause. En effet, notre menu pause crée un léger effet de flou en arrière-plan pour indiquer au joueur qu'il est en pause. En termes simples, un shader peut être ajouté à un matériau pour lui attribuer des attributs très spécifiques. Les shaders sont généralement écrits sous forme de lignes de code et permettent de modifier la couleur, la répartition des pixels et de nombreuses autres fonctionnalités. Dans notre cas, nous modifions la répartition des pixels pour générer l'effet de flou.

Dans le menu pause j'ai rencontré un problème : lorsque l'ordinateur est ouvert, le menu pause peut se lancer en pressant la touche 'P', ce qui crée un conflit et bloque la souris. Pour résoudre ce problème, j'ai ajouté un booléen qui vérifie, à chaque ouverture, s'il est activé, et ainsi le menu pause ne se lance que si les autres éléments ne sont pas activés.

```
private void Update()
{
    if (pickup.OrdiActive == false && pickup.CadenaAcitive == false && pickup.Indice1Active == false)
    {
        if (Input.GetKeyDown(KeyCode.P))
        {
            if (Is_pause == false)
            {
                Is_pause = true;
                pauseMenu.enabled = true;
                Cursor.lockState = CursorLockMode.None;
                playerController.CanMoveHead = false;
            }
            else
            {
                Is_pause = false;
                pauseMenu.enabled = false;
                Cursor.lockState = CursorLockMode.Locked;
                playerController.CanMoveHead = true;
            }
        }
    }
}
```

## Inventaire

Tout d'abord, pour créer mon système d'inventaire dans Unity, j'ai créé un objet vide dans la hiérarchie de Unity que j'ai nommé "Inventaire". C'est le conteneur de tous les éléments de mon inventaire.

Ensuite, j'ai ajouté un script à cet objet pour gérer mon inventaire. Ce script contient une liste d'objets (les éléments de mon inventaire) ainsi que des fonctions pour ajouter, supprimer ou déplacer ces éléments.

Pour représenter les différents éléments de mon inventaire, j'ai créé des objets tels que des potions, des armes et des clés. Ces objets peuvent être des préfabriqués ou des objets que j'ai créés moi-même.

J'ai également ajouté des scripts à ces objets pour qu'ils puissent interagir avec le script d'inventaire. Par exemple, un objet "Lampe" peut avoir un script qui l'ajoute à mon inventaire lorsque je clique dessus.

Ensuite, j'ai placé les objets dans le monde. Lorsque j'interagis avec eux (en cliquant sur "e"), je les ajoute à mon inventaire.

```
1 référence
public void AddItem(ItemData item)
{
    content.Add(item);
    RefreshContent();
}

@ Message Unity | 0 références
public void Start()
{
    inventoryPanel.SetActive(false);
}

@ Message Unity | 0 références
private void Update()
{
    if(!pauseMenu.Is_pause)
    {
        if (Input.GetKeyDown(KeyCode.I))
        {
            //activeSelf permet de savoir l'état de l'inventaire
            inventoryPanel.SetActive(!inventoryPanel.activeSelf);
            //débloquer la souris
            if(Cursor.lockState == CursorLockMode.None)
            {
                Cursor.lockState = CursorLockMode.Locked;
                playerController.CanMoveHead = true;
            }
            else
            {
                Cursor.lockState = CursorLockMode.None;
                playerController.CanMoveHead = false;
            }
        }
    }
}
```

Dans ce script, je gère plusieurs éléments importants. Tout d'abord, je m'assure que l'inventaire ne s'ouvre pas lorsque le joueur est en pause en utilisant un booléen. Ensuite, j'active ou désactive le curseur de la souris. En effet, lorsque le joueur ouvre ou ferme l'inventaire, il doit retrouver le

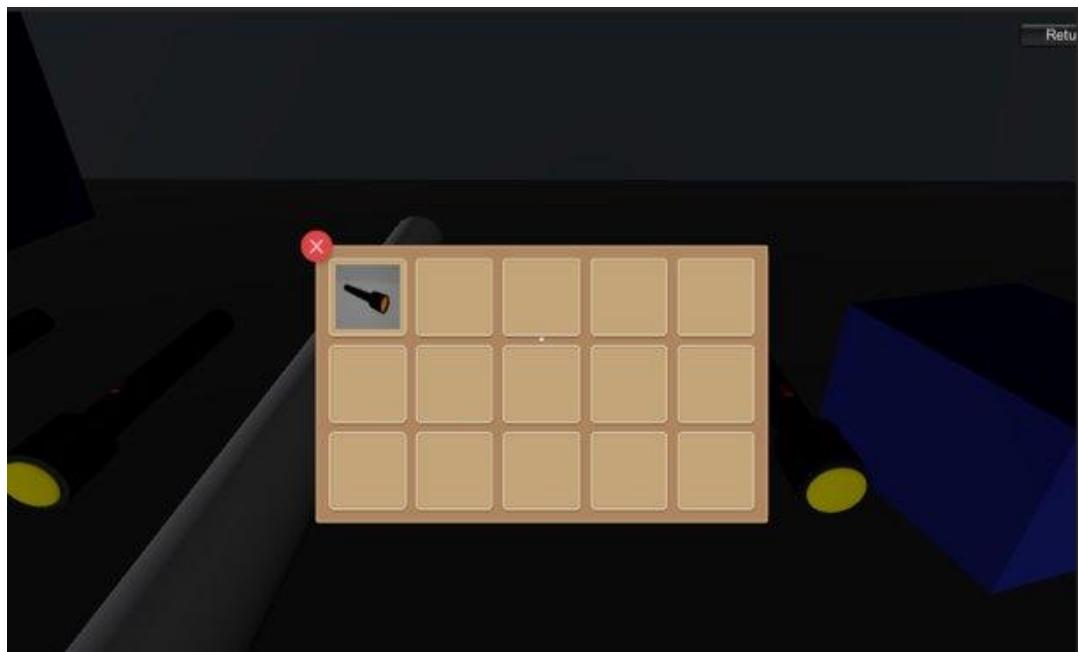
contrôle de sa souris. Ainsi, si le joueur entre ou sort de l'inventaire, j'ajuste l'état de la souris en conséquence.

Pour afficher mon inventaire, j'ai ajouté une interface utilisateur (UI) comprenant une grille de cases. Chaque case représente un emplacement dans mon inventaire. Lorsqu'un objet est ajouté à mon inventaire, il apparaît dans l'une des cases de l'interface.

Ici on vient gérer l'activation ou non de l'UI et des items contenue dedans.

```
0 références
public void closeInventory()
{
    inventoryPanel.SetActive(false);
    Cursor.lockState = CursorLockMode.Locked;
    playerController.CanMoveHead = true;
}

1 référence
private void RefreshContent()
{
    for (int i = 0; i < content.Count; i++)
    {
        inventorySlotsParent.GetChild(i).GetChild(0).GetComponent<Image>().sprite = content[i].visual;
    }
}
```



## Porte et changement de pièce

J'ai mis en place un système de changement de pièce basé sur l'utilisation de tags. Chaque porte de la pièce possède un tag spécifique qui lui est attribué. Lorsque le joueur se trouve devant une porte et appuie sur la touche "E", le jeu détecte si le tag de la porte est présent.

Si le tag de la porte est détecté en même temps que la touche "E" est pressée, le joueur est téléporté vers une position précise dans une autre pièce. Cela permet de créer une transition fluide et instantanée entre les différentes pièces du jeu.

Pour améliorer l'expérience utilisateur, j'ai ajouté une interface utilisateur (UI) spécifique au changement de pièce. Lorsque le joueur s'approche d'une porte, un message ou une icône apparaît à l'écran pour l'inciter à appuyer sur la touche "E" afin d'effectuer le changement de pièce.

Voici les endroits récupérés et mis sous forme de variable.

```
/// <summary>
/// Ici on gère la teleportation du joueur
/// </summary>
private Vector3 InterieurEpita = new Vector3(-73f, 19f, -413f);
private Vector3 Amphi = new Vector3(235, 28, -295);
private Vector3 Etagel = new Vector3(-380f, 37f, -268f);
private Vector3 Etagel2 = new Vector3(-380f, 47f, -268f);
private Vector3 Etagel4 = new Vector3(-380f, 67f, -268f);
private Vector3 intEtagel1 = new Vector3(-90f, 29f, -463f);
private Vector3 intEtagel2 = new Vector3(-90f, 43f, -463f);
private Vector3 intEtagel4 = new Vector3(236f, 27f, -295f);

@ Message Unity | 0 références
public void Start()
{
    InfoIntera.SetActive(false);
    InfoTeleport.SetActive(false);
}
```

Voici une partie de l'utilisation du système de tag.

```
if (Physics.Raycast(transform.position, transform.forward, out hit, PickupRange))
{
    //Portails

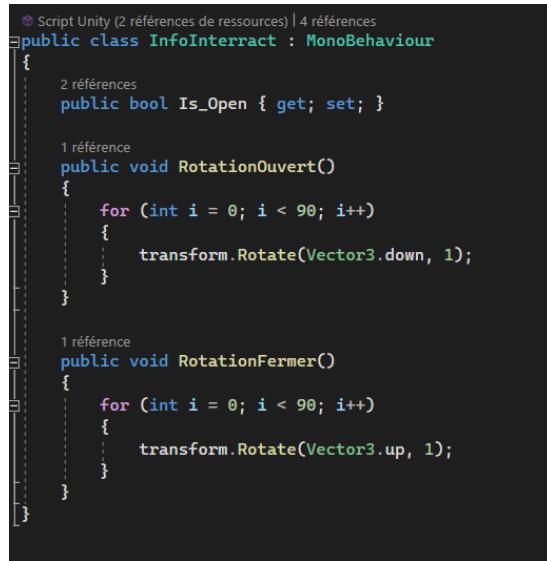
    if(hit.transform.CompareTag("PortailExtEpita"))
        || hit.transform.CompareTag("PortailL")
        || hit.transform.CompareTag("Portail2eme")
        || hit.transform.CompareTag("Portail4eme")
        || hit.transform.CompareTag("PortailInt1")
        || hit.transform.CompareTag("PortailInt2")
        || hit.transform.CompareTag("PortailInt4"))
    {
        InfoTeleport.SetActive(true);

        //Portail ext Epita
        if(hit.transform.CompareTag("PortailExtEpita"))
        {
            if (Input.GetKeyDown(KeyCode.E))
            {
                if(inventory.content.Contains(requireKey))
                {
                    player.transform.position = InterieurEpita;
                    inventory.content.Remove(requireKey);
                }
                else
                {
                    Debug.Log("Pas de clé");
                }
            }
        }
    }
}
```

Dans notre jeu, nous avons mis en place différents systèmes d'utilisation des portes. Outre le système basé sur les tags que j'ai mentionné précédemment, j'ai également implémenté un système d'ouverture de porte manuel.

Ce système permet au joueur de modifier la position de l'objet porte en ajustant spécifiquement sa rotation. Lorsque le joueur interagit avec une porte et appuie sur la touche correspondante, la porte s'ouvre ou se ferme en fonction de son état actuel. Cette mécanique ajoute une dimension interactive et immersive à l'exploration du jeu.

Pour réaliser cela, j'ai utilisé des scripts qui détectent l'interaction du joueur avec la porte et appliquent les transformations nécessaires à sa rotation. Cela permet de simuler de manière réaliste l'ouverture et la fermeture des portes dans le jeu.



```
④ Script Unity (2 références de ressources) | 4 références
public class InfoInteract : MonoBehaviour
{
    2 références
    public bool Is_Open { get; set; }

    1 référence
    public void RotationOuvert()
    {
        for (int i = 0; i < 90; i++)
        {
            transform.Rotate(Vector3.down, 1);
        }
    }

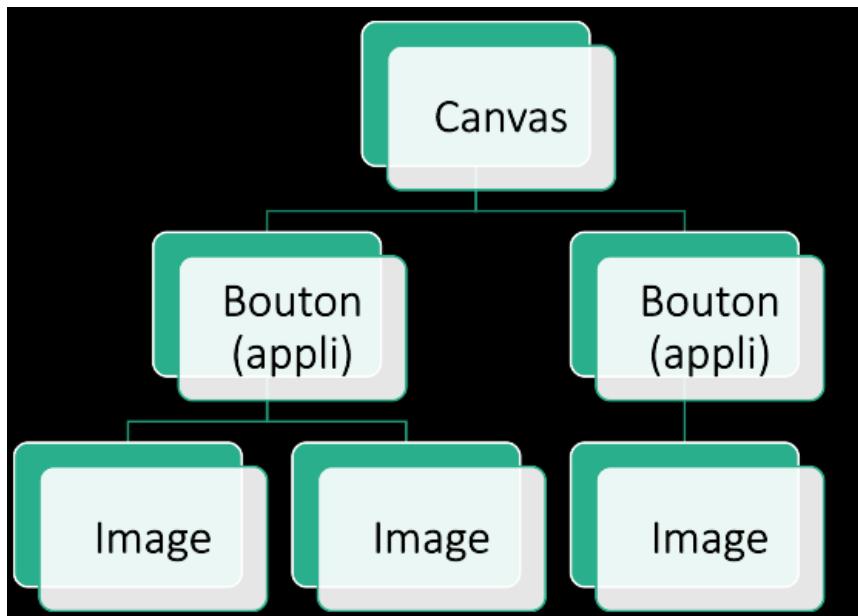
    1 référence
    public void RotationFermer()
    {
        for (int i = 0; i < 90; i++)
        {
            transform.Rotate(Vector3.up, 1);
        }
    }
}
```

En combinant ces deux systèmes d'utilisation de porte, à la fois basé sur les tags et l'ouverture manuelle, nous offrons aux joueurs une expérience de jeu variée et engageante, où ils peuvent interagir avec leur environnement de différentes manières pour progresser dans l'aventure.

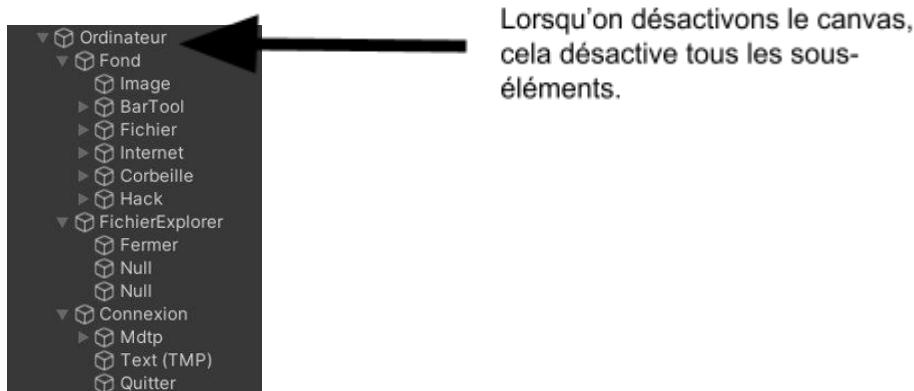
## Simulateur Ordinateur Joueur 2

Pour la création de l'ordinateur, plusieurs éléments devaient être pris en compte. Tout d'abord, il devait être facilement accessible et permettre au joueur de le quitter à tout moment. De plus, il devait comporter un code de déverrouillage et contenir des fichiers et des applications.

Avant d'implémenter cette fonctionnalité, j'ai réfléchi à la méthode la plus efficace. Après quelques tests, j'ai conclu que l'utilisation de canvas était la méthode la plus appropriée. J'ai utilisé un système de hiérarchie entre les différents boutons et fichiers présents dans le canvas. Les boutons peuvent activer d'autres canvas, créant ainsi une structure organisée.

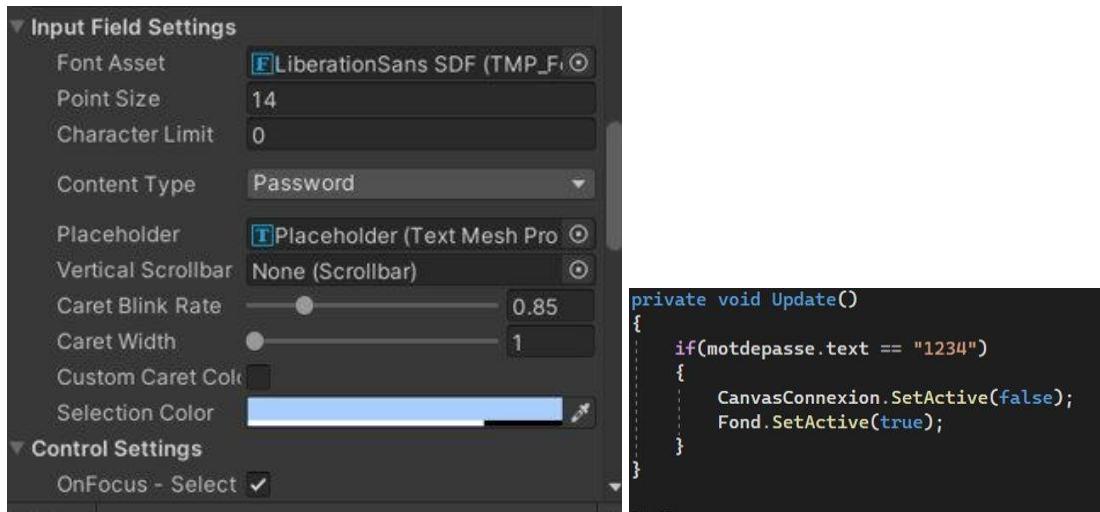


L'efficacité de cette méthode réside notamment dans le fait que, lorsque nous souhaitons sortir de l'ordinateur, il suffit de désactiver le canevas principal. Lorsque nous rouvrons l'ordinateur, les fichiers ouverts précédemment restent accessibles, évitant ainsi d'avoir à saisir à nouveau le mot de passe.



En résumé, l'utilisation de canvas a été la méthode la plus efficace pour créer l'ordinateur du jeu, offrant une expérience utilisateur fluide, pratique et réaliste.

Une fois l'ordinateur réalisé, j'ai abordé la problématique du mot de passe pour le déverrouiller. Unity m'a grandement facilité la tâche grâce à la fonctionnalité des "InputFields". Ce sont des boîtes dans lesquelles on peut entrer du texte, ce texte peut ensuite être comparé à un autre texte pour prendre des actions sur l'ordinateur.



Un autre défi difficile à mettre en place a été l'utilisation de caméras de vidéosurveillance. En effet, le joueur a accès à un système de caméras de surveillance. Voici comment son implémentation s'est déroulée. Tout d'abord, j'ai dû imaginer un moyen de passer d'une caméra à une autre. Cependant, étant donné que le jeu est multijoueur, cette tâche était trop lourde et complexe. Le jeu ne savait pas quel joueur devait prendre possession de quelle caméra, ce qui a entraîné de nombreux conflits.

Finalement, j'ai décidé de l'implémenter de la manière suivante : lorsque le joueur entre dans l'application de caméra, nous téléportons la caméra du joueur à des endroits spécifiques. Ainsi, en enregistrant les coordonnées de chaque emplacement des caméras, nous pouvons assez simplement recréer l'effet de vidéosurveillance.

Ici on utilise le type vector3 pour la position et Quaternion pour la rotation.

```
Vector3 BackUpPosCam;
Quaternion BackUpRotCam;

Vector3 PosCamSousSol = new Vector3(-43.2f, 0.4f, -347);
Quaternion RotCamSousSol = new Quaternion(2.3f, -227.2f, 0, 180);

Vector3 PosCamRDC1 = new Vector3(-16, 22, -460);
Quaternion RotRDC1 = new Quaternion(2.3f, -51, 0, 180);

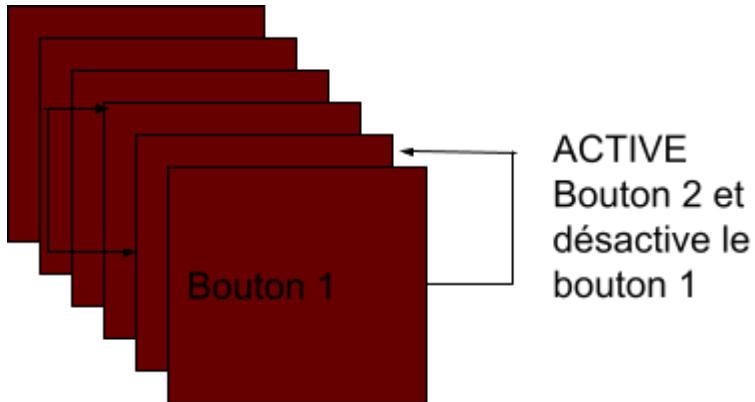
Vector3 PosCamRDC2 = new Vector3(-90, 25.38f, -431);
Quaternion RotRDC2 = new Quaternion(16f, 88, 0, 180);

Vector3 PosCamLer = new Vector3(-95, 40, -442);
Quaternion RotLer = new Quaternion(18.5f, 53, 0, 180);

Vector3 PosCam2eme1 = new Vector3(-69.5f, -49.5f, -447);
Quaternion Rot2eme1 = new Quaternion(18.5f, -47, 0, 180);

Vector3 PosCam2eme2 = new Vector3(-69.5f, 49.5f, -406);
Quaternion Rot2eme2 = new Quaternion(18.5f, -153, 0, 180);
```

Pour faciliter le changement de caméra, j'ai mis en place 6 boutons différents qui se chevauchent. Chaque bouton est associé à un code spécifique qui lui permet d'activer sa propre caméra. Voici un schéma explicatif :



Lorsqu'un joueur presse le bouton 1, celui-ci se désactive et le bouton 2 s'active à sa place. Ce processus se répète pour chaque bouton successif, permettant ainsi une transition fluide entre les différentes caméras.

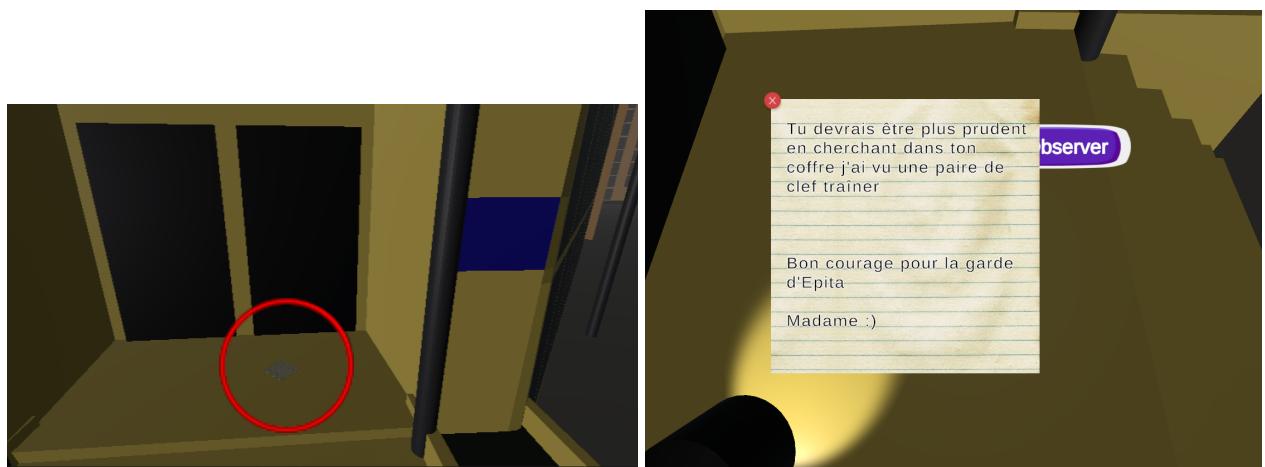
## Première Énigme

La première énigme se déroule de la manière suivante : le joueur se trouve devant Epita et doit se rendre devant la porte. Lorsqu'il s'approche, un message s'affiche à l'écran lui indiquant que la porte est fermée. Pour réaliser ce message, nous utilisons une zone de texte située en bas du canvas du personnage. Lorsque le personnage se présente devant la porte sans avoir la clé, nous remplissons cette zone de texte avec le message souhaité. Cela fonctionne comme un conseil donné au joueur.

```
//active message pour intéragir
InfoIntera.SetActive(true);

if (hit.transform.CompareTag("Destroy"))
{
    //Objet à casser
    if (Input.GetKeyDown(KeyCode.E))
    {
        if (inventory.content.Contains(requirePiedBiche))
        {
            //on détruit le gameobject car ici on souhaite détruire la vitre de la voiture du gardien
            Destroy(hit.transform.gameObject);
        }
        else
        {
            //ManqueKey est la zone de texte vide qui est rempli pour aider le perso
            ManqueKey.text = "La voiture est fermée ...";
        }
    }
}
```

Un autre système d'indices est représenté par les feuilles disposées par terre. Elles prennent la forme d'objets qui affichent une image à l'écran du personnage. Le joueur doit regarder la feuille pour en lire le contenu. Pour mettre en place ce système, nous utilisons les raycasts et les interfaces utilisateur (UI) pour détecter si le joueur regarde l'image et s'il appuie sur la touche "E".



Le joueur devra ensuite comprendre qu'il doit se diriger vers la voiture du gardien. En observant le script ci-dessus, on remarque que le système est assez simple. Il suffit de cibler la vitre de la voiture et, dès que le joueur possède l'objet "pied de biche", il est en mesure de la briser.

Du côté du joueur dans sa chambre, une nouvelle fonctionnalité est utilisée : le rigidbody. En ajoutant un rigidbody à un objet, on crée l'effet de l'objet tombant sur le sol. Dans ce cas, cet effet sera utilisé lorsque le joueur utilise une visseuse sur le tableau présent dans sa chambre.

Afin d'ajouter du réalisme lorsque nous ajoutons ce rigidbody, nous lui appliquons une force très légère. Ainsi, derrière le tableau, le joueur découvre un rébus qu'il doit résoudre.

```
private Vector3 forceTableau = new Vector3(1, 0, 0);

public bool firstTime = false;
public bool Fall = false;

[SerializeField]
Visse visse1;

[SerializeField]
Visse visse2;

[SerializeField]
Visse visse3;

[SerializeField]
Visse visse4;

④ Message Unity | 0 références
private void Update()
{
    if(firstTime == false && visse1.isVisser == false && visse2.isVisser == false
       && visse3.isVisser == false && visse4.isVisser == false)
    {
        this.transform.gameObject.AddComponent<Rigidbody>();
        this.transform.gameObject.GetComponent<Rigidbody>().mass = 2;
        this.transform.gameObject.GetComponent<Rigidbody>().AddForce(forceTableau);
        this.transform.gameObject.GetComponent<BoxCollider>().enabled = false;
        firstTime = true;
    }
}
```

## Deuxième Énigme

La deuxième énigme est une énigme de coopération. Le joueur se trouve plongé dans l'obscurité à Epita et découvre une lampe qui lui permettra de s'orienter et de récupérer les éléments nécessaires pour progresser dans le jeu. L'objectif est de trouver la bonne combinaison pour un disjoncteur à trois positions. À Epita, le joueur dispose de 20 fichiers comportant des disjoncteurs différents. Ils devront collaborer pour trouver le bon. Pour réaliser cette combinaison, j'ai décidé d'implémenter deux booléens, "On" et "Off", pour chacun des trois disjoncteurs.

```
public class Disjoncteur : MonoBehaviour
{
    public bool On = false;
    public bool Off = true;

}
```

Ces éléments sont très utiles, car ensuite, je n'ai qu'à vérifier si la bonne combinaison est respectée à chaque mise à jour du jeu en faisant référence à mes trois disjoncteurs.

```
public class SetupDisj : MonoBehaviour
{
    //Pour résoudre l'énigme il faut que les deux derniers interrupteurs soient désactivée.

    [SerializeField]
    Disjoncteur D1;

    [SerializeField]
    Disjoncteur D2;

    [SerializeField]
    Disjoncteur D3;

    [SerializeField]
    List<Light> lights;

    [SerializeField]
    GameObject Bloquage;
    // Message Unity | 0 références
    private void Start()
    {
        foreach (Light light in lights)
        {
            light.enabled = false;
        }
    }

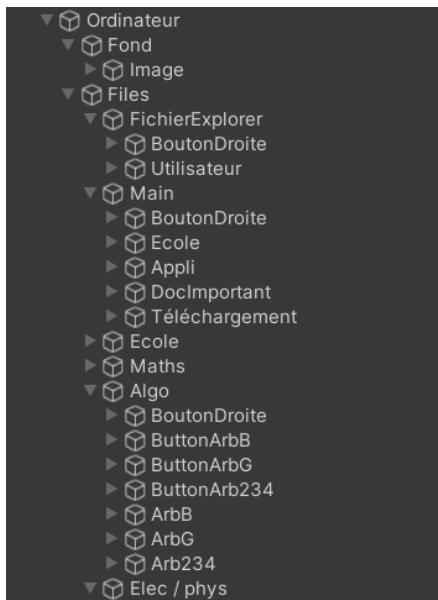
    // Message Unity | 0 références
    private void Update()
    {
        if (D1.On == false && D2.On == true && D3.On == true)
        {
            foreach (Light light in lights)
            {
                light.enabled = true;
            }
            Bloquage.SetActive(false);
        }
    }
}
```

On remarque également que je dois m'occuper de la gestion du disjoncteur, qui permet d'allumer toutes les lumières d'Epita. J'ai donc créé une liste de Light qui comprend toutes les lumières d'Epita. Dans un premier temps, lors du lancement du jeu, je les éteins toutes. Ensuite, lorsque la bonne combinaison est entrée, je les allume. Il est important de noter que pour éviter que le joueur ne se promène dans les autres étages, ces derniers sont bloqués par un mur invisible. Lorsque le joueur s'approche trop de ce mur, un indice s'affiche lui indiquant de se diriger vers une des quatre directions.

Afin de rendre les boutons réalistes, il était nécessaire de trouver un moyen de changer leur état visuel. Pour ce faire, j'ai utilisé la fonction Rotate de Unity, qui nous permet d'effectuer une rotation de l'angle souhaité autour d'un axe donné. On retrouve toujours l'utilisation du Raycast.

```
//On gère le mouvement des disjoncteurs
if (hit.transform.CompareTag("Disjoncteur"))
{
    if(Input.GetKeyDown(KeyCode.E))
    {
        hit.transform.Rotate(Vector3.up, 180);
        if (hit.transform.gameObject.GetComponent<Disjoncteur>().On)
        {
            hit.transform.gameObject.GetComponent<Disjoncteur>().On = false;
            hit.transform.gameObject.GetComponent<Disjoncteur>().Off = true;
        }
        else
        {
            hit.transform.gameObject.GetComponent<Disjoncteur>().On = true;
            hit.transform.gameObject.GetComponent<Disjoncteur>().Off = false;
        }
    }
}
```

Du côté du joueur n°2, nous utilisons le système expliqué pour l'ordinateur. En poussant ce système, nous obtenons un arbre complexe et chargé. Voici un résumé des problèmes auxquels j'ai été confronté : tout d'abord, lors du parcours des fichiers, il est possible de revenir en arrière, ce qui signifie que chaque bouton "revenir en arrière" doit être codé. Cela a rapidement conduit à un grand nombre de fonctions et m'a rendu difficile la navigation dans le code. De plus, l'ordinateur doit contenir des fichiers inutiles qui serviront de leurre pour le joueur. Cependant, implémenter ces fichiers nécessite de répéter le même processus à chaque fois. J'aurais dû me préparer à cela en essayant d'optimiser une fonction capable de fermer l'objet ouvert. Or j'ai simplement codé chaque bouton "quitter le fichier" en spécifiant le fichier à désactiver.



Le disjoncteur débloque l'accès au premier étage. Pour ce faire, je désactive un cube invisible lors de l'activation de ce dernier.

## Troisième Énigme

Dans cette troisième énigme, nous réutilisons les fonctionnalités précédemment utilisées. Le joueur accède au premier étage et doit trouver un bouton caché derrière la machine à café. Pour trouver le bouton, l'autre joueur a accès à Internet où il trouve une page sur le café. Ce bouton ouvre une porte secrète au niveau des escaliers. Nous gérons cela en utilisant des vecteurs et des quaternions : lorsque le bouton est activé, nous positionnons la porte au bon endroit.

La clé USB est cachée dans les casiers, et les joueurs doivent s'aider mutuellement : l'un a accès aux numéros de casier et au mot de passe, tandis que l'autre a accès aux casiers eux-mêmes. Pour ce faire, j'utilise une fois de plus la fonctionnalité d'"Input Field", qui permet de vérifier si le mot de passe est correct. Les autres casiers ont également des mots de passe, mais ils ne peuvent pas s'ouvrir. Une fois que la clé a été récupérée, le joueur doit introduire la clé dans les serveurs situés dans la pièce secrète. Ici, nous jouons avec la possibilité de cacher et de montrer un objet : lorsque le joueur appuie sur la touche 'E' avec la clé USB vers les serveurs, l'objet clé USB apparaît, donnant ainsi l'impression que le joueur insère réellement une clé.

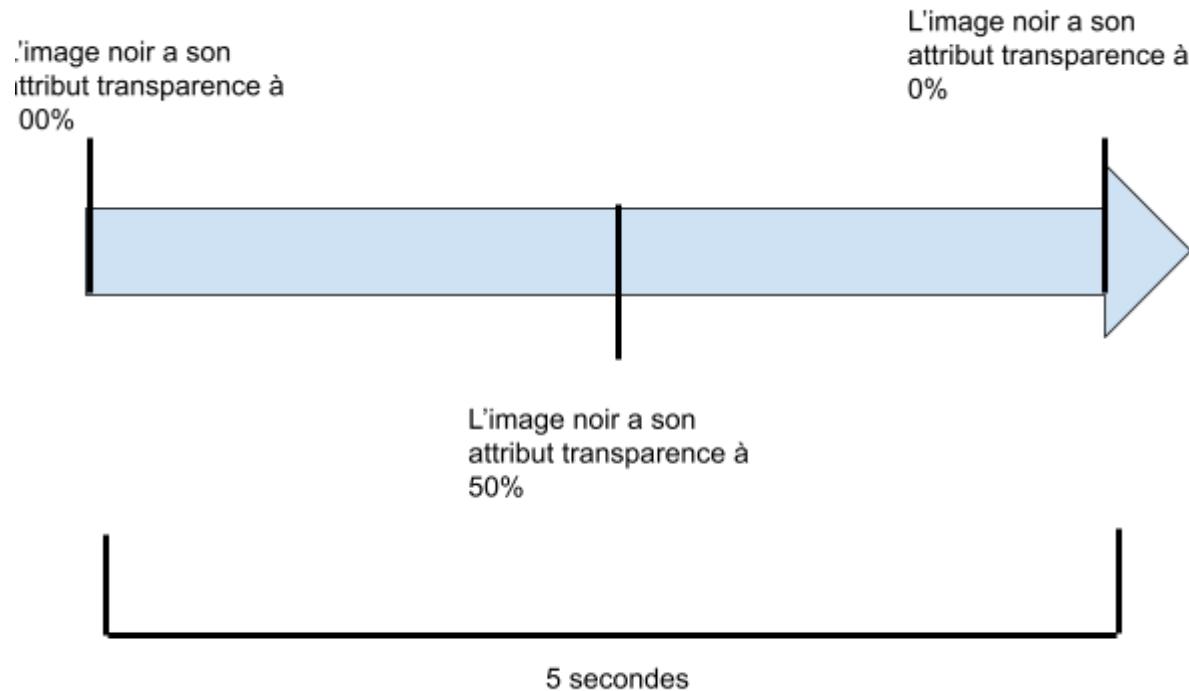
## Quatrième Énigme

Ici, nous utilisons une nouvelle fonctionnalité d'Unity appelée les animateurs. L'énigme se présente de la manière suivante : le joueur se trouve devant les bureaux de Monsieur Saber et de Madame Fatoumata dans Epita. Il doit choisir entre les deux portes pour continuer à avancer. Le joueur doit observer à travers la fenêtre des bureaux et choisir celui où il n'y a personne (le bureau de Madame Fatoumata).

Si le joueur entre dans le bureau de Monsieur Saber, il se retrouvera figé devant la porte et verra l'animation réalisée par Fabien se déclencher. Afin de gérer cet événement, j'utilise un contrôleur d'animation qui enchaîne l'animation de Monsieur Saber se levant et se déplaçant. Ces deux événements sont contenus dans une animation. Grâce à cela, nous pouvons lancer l'animation à partir d'un événement.

Je récupère le temps nécessaire pour que les animations se déroulent, j'attends et je fais perdre au joueur. Lorsque le joueur perd, notre objectif n'est pas de le faire recommencer depuis le début. Son écran se noircit simplement, la souris se débloque et il a la possibilité d'apparaître au début du jeu, soit à l'extérieur d'Epita.

Pour noircir progressivement l'écran du joueur, nous utilisons à nouveau les animations. Une animation peut être visualisée comme une flèche chronologique :



Une fois entré dans le bureau de Madame Fatoumata, le joueur sera accompagné de son coéquipier, qui aura accès à des archives vidéo de Madame Fatoumata (réalisées par Fabien). Parmi ces vidéos, l'une d'entre elles révèlera Madame Fatoumata en train de cacher une paire de clés dans un boîtier. Le joueur à Epita devra récupérer ces clés, monter à l'étage et, en raison du manque de temps, nous terminerons par une salle à ouvrir contenant les examens finaux. Le joueur n'aura plus qu'à sortir de cette salle, et la partie sera gagnée. Les deux joueurs se retrouveront ensuite téléportés dans une salle où des feux d'artifice célébreront leur succès dans le jeu.

#### Conclusion personnel :

Ce jeu a été un véritable défi pour moi, me permettant de découvrir de nombreuses fonctionnalités et aspects de Unity. Je sens une réelle progression dans mes compétences techniques grâce à cette expérience. Malgré quelques événements inattendus au sein de l'équipe et les obligations de reprendre le travail de certains membres, j'ai su persévérer et donner forme à un jeu. Cela a nécessité de l'effort et de la détermination, mais le résultat en valait la peine. Cette expérience m'a également enseigné l'importance de la flexibilité et de l'adaptabilité face aux imprévus. J'ai appris à gérer les défis et les obstacles qui se sont présentés, et cela m'a rendu plus résilient en tant que développeur. Je suis fier du travail accompli malgré les difficultés rencontrées, et je suis reconnaissant d'avoir eu l'opportunité de progresser dans mon domaine. Cela m'a également permis de renforcer ma capacité à travailler en équipe et à collaborer efficacement, des compétences essentielles pour mes futurs projets.

## Récit de la réalisation

### Les joies

Nous, l'équipe d'Atipe, composée de quatre personnes passionnées, avons vécu une expérience extraordinaire dans la réalisation de notre jeu vidéo. Malgré quelques moments de relâchement de certains membres, notre cohésion au sein du groupe a été excellente, renforçant ainsi notre détermination commune à réussir.

La création de ce jeu à quatre mains a démontré avec éclat que l'union fait la force. Chacun a pu apporter sa touche personnelle, son expertise et ses idées innovantes, faisant ainsi émerger une véritable synergie créative. Ensemble, nous avons repoussé les limites de notre imagination, empruntant des chemins inexplorés.

Le format du jeu vidéo a été un choix parfaitement adapté pour notre équipe. Il nous a offert une totale liberté artistique et un vaste champ d'expression pour laisser libre cours à notre créativité débordante. Les frontières de l'imaginaire se sont estompées, nous permettant de faire prendre vie à nos idées les plus audacieuses dans un univers virtuel captivant.

La découverte du logiciel Blender a été une révélation pour notre groupe. Ce puissant outil de création 3D nous a ouvert de nouvelles perspectives et nous a permis de donner vie à des personnages, des paysages et des scènes d'une qualité époustouflante. Les nombreuses heures passées à expérimenter et à maîtriser ce logiciel ont été gratifiantes et ont renforcé notre amour pour l'art numérique.

Au fil des mois, nous avons pu ressentir une profonde joie et satisfaction à chaque étape franchie dans la réalisation de notre jeu. Des moments d'émerveillement devant des graphismes saisissants, des fous rires partagés lors de nos séances de brainstorming, des bonds de joie à chaque bug résolu, toutes ces petites victoires ont contribué à créer un sentiment d'accomplissement collectif. Nous nous sommes soutenus mutuellement, avons surmonté les obstacles ensemble et avons célébré chaque succès comme une victoire de l'ensemble du groupe. La réalisation de "Atipe" a été bien plus qu'un simple projet de jeu vidéo. C'était une aventure humaine, un voyage au cœur de notre créativité et de notre amitié.

## Les peines

Notre équipe avait de grandes ambitions lorsqu'elle s'est lancée dans le développement d'un jeu vidéo passionnant il y a six mois. Au début, la répartition des tâches semblait prometteuse, avec chacun ayant une responsabilité spécifique. Cependant, les peines ont rapidement commencé à s'accumuler.

Malheureusement, certains membres de l'équipe n'ont pas réussi à accomplir leurs tâches assignées, ce qui a eu un impact considérable sur l'avancement global du projet. Les autres membres de l'équipe ont dû prendre en charge ces tâches non réalisées, ce qui les a obligés à négliger leurs propres travaux et à prendre du retard. La frustration s'est emparée de nous face à cette situation.

De plus, le nombre de réunions aurait dû être augmenté pour permettre à chacun de se mettre à jour et de discuter des problèmes rencontrés. Les membres de l'équipe auraient ainsi pu mieux se coordonner et trouver des solutions plus rapidement. Malheureusement, le manque de communication a aggravé les problèmes existants.

Un autre obstacle majeur qui a freiné considérablement l'avancement du projet a été les problèmes rencontrés avec Git, le système de contrôle de version utilisé. Les conflits de fusion, les erreurs de synchronisation et les pertes de données ont créé des retards importants et ont entraîné une perte de confiance dans l'outil.

Malgré son dévouement et son enthousiasme initial, notre équipe s'est retrouvée confrontée à des peines incessantes tout au long du processus de développement du jeu. La déception de voir les tâches non accomplies par certains membres, combinée aux retards causés par les problèmes de coordination et les difficultés techniques, a eu un impact démoralisant sur l'équipe.

Cependant, malgré ces épreuves, nous sommes restées unies et déterminées à surmonter ces obstacles. Nous avons pris conscience de l'importance d'une communication claire et régulière, ainsi que de l'optimisation des processus de travail. Les membres de l'équipe se sont engagés à résoudre les problèmes de Git et à mettre en place des mesures pour éviter que les tâches ne soient négligées à l'avenir.

## **Aspect économique :**

Le projet aura un faible coût, une estimation de ce budget serait entre 00 - 100 €, car nous souhaitons faire nous-mêmes tout ce qui est en notre mesure. En effet, si notre projet fonctionne nous pourrions l'implémenter dans Steam (logiciel de Partage de jeu), ce qui devrait coûter 100\$. De plus, nous pourrions potentiellement trouver des Shaders vendu autour des 1-5\$. Et pour finir de nombreuses formations de Unity sont vendues sur des sites comme Udemy ou OpenClassroom.

## **Conclusion :**

La conclusion de ce projet de 6 mois se résumerait par une expérience enrichissante à la fois sur le plan humain et technique. Travailler en équipe de trois sur un logiciel dont nous ne connaissons pas l'existence, tout en jonglant avec les difficultés liées à nos cours, a été un défi stimulant. Malgré les contraintes, nous sommes fiers du rendu final, même s'il n'est peut-être pas totalement exhaustif. Cette expérience nous a permis de développer nos compétences techniques, d'apprendre à collaborer efficacement et de surmonter les obstacles rencontrés. Nous avons acquis une meilleure compréhension des processus de développement de logiciels et avons renforcé notre capacité à travailler en équipe. Au-delà des connaissances techniques, cette expérience nous a également appris l'importance de la persévérance, de la communication et de l'adaptabilité. Nous sommes reconnaissants d'avoir eu cette opportunité de croissance personnelle et professionnelle, et nous sommes impatients d'appliquer les leçons apprises dans nos futurs projets.