Rapport de Soutenance

Louis POURRAT
Hugo BLANCHET
Rayane TIBAR
Martin BRUCHE

Janvier 2025

Table des matières

1 Introduction

1.1 Présentation du projet

Notre projet s'intitule Lurking Home, c'est un jeu d'horreur et d'escape où l'objectif principal est de s'échapper d'une maison mystérieuse. Les joueurs doivent accomplir des tâches dans chaque pièce pour obtenir des fragments d'âme. Une fois ces fragments rassemblés, ils permettent de créer un artefact puissant, capable de libérer la maison de son emprise maléfique. Cependant, le temps est compté : l'entité qui hante la maison cherche à capturer les joueurs, et la maison elle-même semble prête à les engloutir à tout instant. Si les joueurs échouent à compléter l'artefact à temps, ils seront pris au piège et la maison les cousumera jusqu'à ce qu'une nouvelle équipe tente sa chance. Ce jeu mêle exploration, accomplissements de tâches et tension constante, tout en plaçant les joueurs dans une course contre la montre pour leur survie.

1.2 Présentation des membres

- Louis POURRAT (Chef de projet)

À la suite de mon année de terminale avec les spécialités Maths et NSI, j'ai décidé d'aller à l'EPITA et j'en suis arrivé à réaliser un projet de jeu vidéo en groupe pour le second semestre.

La programmation est un domaine que j'affectionne particulièrement grâce à son aspect créatif qui permet, à partir de rien, de créer des logiciels, des applications ou même des jeux.

Lors de mon année de terminale, par groupe de trois, nous avons réalisé un petit projet de jeu durant lequel nous avons recréé le jeu vidéo Pac-Man. Ce projet m'a permis de développer mes compétences en Python, d'apprendre à travailler en groupe, mais surtout de comprendre que la rédaction d'un cahier des charges est indispensable pour un projet.

Au sein de l'équipe, je vais pouvoir apporter mes compétences en développement web, qui sont plutôt solides grâce à mes pratiques personnelles. De plus, j'ai une assez bonne connaissance des concepts globaux de la programmation, grâce à mon expérience personnelle mais également scolaire, ce qui va me permettre d'être efficace pour le projet.

- Rayane TIBAR

Je m'appelle Rayane TIBAR, 18 ans, né au Maroc et je suis passionné pour la technologie dès mon plus jeune âge.
À 14 ans, j'ai construit mon premier ordinateur, apprenant la patience et l'ingéniosité nécessaires pour assembler un système complet.
À 16 ans, j'ai réalisé un projet en Python inspiré des mosaïques marocaines, combinant récursivité et géométrie pour créer des motifs numériques reflétant mon héritage culturel.
Arrivé en France à 18 ans pour étudier à l'EPITA, une école d'ingénieurs en informatique j'étais prêt à pousser cette passion pour l'informatique encore plus loin.

- Hugo BLANCHET

Je m'appelle Hugo BLANCHET, j'ai 17 ans et je suis étudiant à l'EPITA. J'ai rejoint l'EPITA pour perfectionner mes compétences en informatique et devenir ingénieur dans ce domaine.

Au lycée, dans le cadre de la spécialité NSI, j'ai réalisé un projet de jeu vidéo en Python. Avec mon camarade Louis, nous avons recréé notre version du jeu Pac-Man. Ce projet m'a permis d'acquérir de l'expérience en gestion de projet et en programmation.

Je pense que cette expérience acquise grâce à ce projet me sera utile pour notre projet Lurking Home, car elle m'a permis de mieux comprendre comment visualiser et gérer le travail d'équipe sur un projet commun. Elle m'a aussi donné les bases de la programmation d'un jeu vidéo et un aperçu de ce que représente la création d'un jeu vidéo.

J'ai hâte de réaliser un nouveau jeu vidéo qui, je l'espère, surpassera tout ce que j'ai pu accomplir jusqu'à présent.

- Martin BRUCHE Depuis mon adolescence, l'informatique m'a toujours fasciné, en particulier le développement de jeux vidéo. Cette passion m'a conduit à poursuivre des études à l'EPITA, où je continue de développer mes compétences.

Au lycée, j'ai travaillé sur des projets concrets comme la création d'un site internet sur les sneakers en groupe, ce qui m'a permis d'apprendre la conception web et la gestion d'équipe.

J'ai également acquis de l'expérience en vente en ligne sur Vinted, où j'ai développé des compétences en communication et en gestion de clients. Mon parcours s'est enrichi grâce à un stage d'un mois chez W3COM, qui m'a permis de renforcer mes compétences techniques, et par une expérience de sept semaines en grande surface, où j'ai développé des compétences en gestion et en service client.

1.3 Rappels sur les tâches individuelles et communes

	Louis	Rayane	Hugo	Martin
Intelligence artificielle				
Décors et joueurs design				
Site web				
Multijoueur				
Logo et jacket				
Bande sonore				
Gameplay				
Histoire et DA				
Conception des missions				

	Responsable
	Suppléant

Nous avons légèrement modifié notre répartition des tâches depuis le cahier des charges, nous avons mis seul Louis pour le multijoueur et Hugo pour l'IA alors qu'il y'avait Martin en assistant normalement pour les deux, en contre partie nous avons enlevé Louis de la responsabilité du site web et Martin l'a remplacé.

	1re Soutenance	2de Soutenance	3e Soutenance
Intelligence artificielle	25%	50%	100%
Décors et joueurs design	25%	50%	100%
Site web	15%	50%	100%
Multijoueur	35%	70%	100%
Logo et jacket	50%	100%	100%
Bande sonore	15%	70%	100%
Gameplay	15%	40%	100%
Histoire et DA	75%	100%	100%
Conception des missions	30%	60%	100%

Voici notre planning pour notre projet, nous allons donc voir si nous avons l'avons respecté pour cette première soutenance.

2 Avancements du projet

2.1 Histoire

Concernant l'histoire, nous l'avons rapidement eu en tête et nous avons réussi à l'enrichir rapidement et nous pensons qu'elle est quasiment complète donc nous avons bien respecté cette objectif, il reste cependant quelques détails à corriger.

Synopsis général :
Tu te réveilles dans une maison lugubre, sans souvenir de comment tu es arrivé là. Une entité rôde, imprévisible, prête à frapper à tout moment. La maison semble vivante, chaque pièce évoquant une émotion troublante ou un traumatisme. Ton objectif est simple : explorer, résoudre les énigmes de chaque pièce, et échapper à la maison avant qu'elle ne te consume... ou avant que l'entité ne t'attrape.

Concept central du lore :
La maison est un reflet de l'esprit brisé des âmes d'une famille piégée. L'entité est une projection de ces âmes, alimentée par la douleur et les peurs qui hantent les pièces. Les joueurs incarnent des personnages qui, pour des raisons mystérieuses, sont connectés aux traumatismes représentés dans la maison. Leur survie dépend de leur capacité à affronter et surmonter ces peurs.

Chaque pièce révèle subtilement des fragments de l'histoire de l'entité et de son lien avec les joueurs, mais rien n'est explicitement dit. Les indices doivent être glanés à travers des détails visuels, des notes, et des éléments interactifs.

Sortir de la maison :
Les joueurs doivent collecter des "fragments d'âme" (ou un équivalent) dans chaque pièce. Une fois assemblés, ils reconstituent un artefact qui "libère" la maison. Si les joueurs échouent à temps, ils meurent, et la maison change de forme, prête à accueillir ses prochaines victimes.

2.2 Gameplay

2.2.1 Déplacements et actions

Notre joueur peut se déplacer librement sur une surface plate avec les touches Z,Q,S,D. Pour ce faire, nous récupérons les inputs du joueur sur chaque axe (l'axe horizontale et l'axe verticale). Sur chaque axe les valeurs peuvent être 1 ou -1, si la touche Z est pressé la valeur sera 1 donc positive, le joueur avancera et si le joueur presse la touche S la valeur sera -1 donc négative, le joueur reculera. C'est la même chose pour l'axe horizontale avec les touches D et Q respectives.

Ensuite on va utiliser un Vector3 qui permet de renvoyer la distance entre deux points en prenant compte les 3 trois axes (x,y,z). Les deux Vector3 pour les deux axes respectifs permettent de connaitre les directions choisies par le joueur. Puis, on va regrouper ces directions en un seul Vector3 qu'on va multiplié par la speed de base de notre joueur pour obtenir la vitesse réelle sur le jeu.

Enfin, on va utiliser une méthode du RigidBody (comportement physique du joueur) qui s'appelle MovePosition en multipliant la vitesse réelle calculée précedemment avec une propriété qui représente l'intervalle de temps entre deux appels consécutifs de la méthode FixedUpdate (méthode utilisé pour les calculs liés à la physique et appelé à intervalle réguliers)

```
private void PerformMovement()
{
    if (velocity != Vector3.zero)
    {
        rb.MovePosition(rb.position + velocity * Time.fixedDeltaTime);
    }
}
```

Extrait du script qui permet de changer la position du joueur

De plus, notre joueur a la possibilité de marcher plus vite. Il suffit simplement de créer un booléen qui vérifie si la touche MAJ est enfoncée et si c'est true alors la speed sera multiplié par une variable runSpeed afin que le joueur se déplace plus rapidement tant qu'il appuie sur MAJ.

Pour information, dans notre jeu le joueur ne pourra ni sauter ni s'accroupir donc c'est pour cela que ces mécaniques ne sont/seront pas implémentés.

Pour le gameplay, les deux prochains objectifs majeurs vont être de créer le système de pick-up des objets mais également de gérer les collisions avec l'environnement.

2.2.2 Mouvements caméra

Notre joueur a également la possibilité de bouger la sa caméra comme il le souhaite. Cela marche un peu comme les déplacements, cependant l'axe X de la souris et l'axe Y de la souris ne vont pas être gérés de la même façon.

Pour ce qui est de l'axe X de la souris il suffit de récupérer la valeur de la rotation effectué, ce qu'il faut savoir c'est que l'axe Y de la rotation du joueur correspond à un regard sur la gauche donc à l'axe X pour notre souris, c'est pour cela que l'axe X de la souris correspond à l'axe Y du joueur. Une fois la valeur récuperé, on va ensuite créer un Vector3 comme pour les déplacements pour obtenir la rotation sur l'axe X de la souris en le multipliant par la sensibilité de la souris défini dans une variable au préalable.

Comme pour les déplacements, on va appelé une méthode du RigidBody qui s'appelle MoveRotation et qui prend en paramètre un Quaternion (correspond à la représention d'une rotation), ce Quaternion passé en paramètre va être la multiplication de la position actuelle de notre RigidBody avec la rotation qu'on a calculée au préalable.

Pour l'axe Y de la souris, on ne peut pas directement agir sur le RigidBody car si on modifie l'axe X du RigidBody, le physique du joueur va se pencher en avant ou en arrière et c'est pas ce qu'on veut, on veut simplement orienter sa vision. Donc au lieu

de modifier la rotation de tout le joueur on va modifié l'inclinaison de la caméra du joueur.

Pour ce qui est du code, c'est quasiment le même principe que pour l'axe X de la souris, le seul changement principal est le fait qu'on utilisera la variable correspondant à la caméra du joueur et non à son RigidBody.

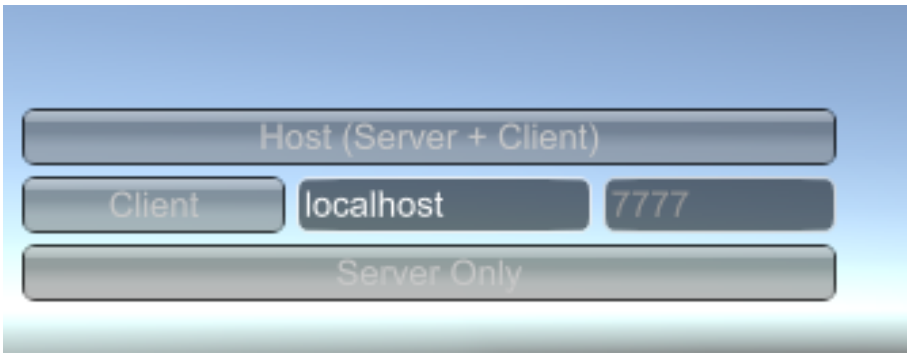
2.3 Multijoueur

Le multijoueur a été une des priorités car nous voulons absolument l'intégrer dans notre jeu le plus rapidement possible. En effet, nous ne voulons pas que dès que notre jeu soit fini donc jouable en solo, le multijoueur perturbe le bon fonctionnement du jeu et que nous devons recommencer certaines étapes, ce serait une perte de temps. C'est pour cela que nous avons plutôt bien avancer sur cette tâche.

2.3.1 Mise en place en local

Nous avons donc réussi à mettre en place un système en réseau local grace à l'asset Mirror disponible gratuitement dans l'asset store de Unity.

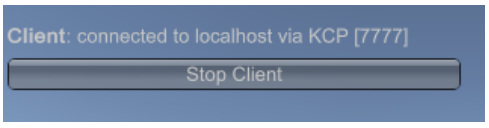
Comme nous sommes en local, nous avons besoin de faire un build, qui va nous permettre d'avoir un éxecutable de notre projet pour avoir notre jeu sur deux fenêtres différentes et donc se faire rejoindre 2 instances de notre prefab du joueur sur la même machine.



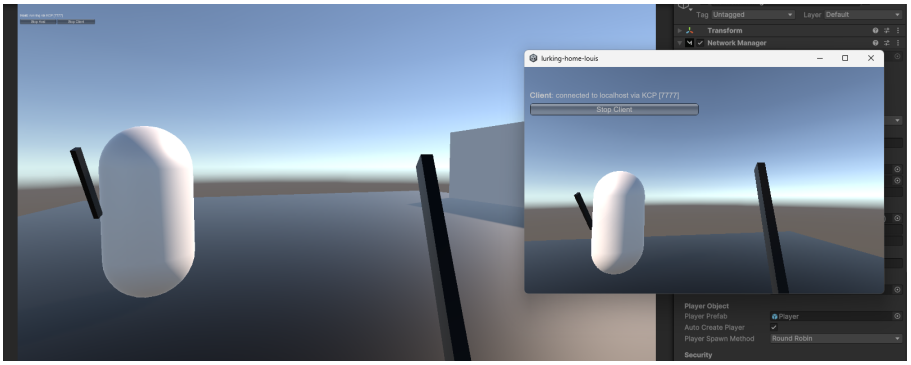
Interface permettant de se connecter au serveur local

Le fonctionnement du multijoueur en local est assez simple, un des joueurs se connecte au serveur en cliquant sur Host et après chaque joueur (en fonction du nombre de build ouvert sur la machine) clique sur Client pour rejoindre l'hôte.

L'host est bien connecté



Le client est bien connecté



Une fois les deux instances connectés, nous pouvons bien observer nos deux joueurs respectifs qui se voient bien.

Le multijoueur est gérer par un GameObject vide intitulé NetworkManager. Tout d'abord, on a du ajouté component appelé KCP Transport qui permet de gérer les possibles pertes de paquets avec une meilleure gestion de la latence.

A partir de ce moment la, nous avions deux joueurs qui se voyait mais les mouvements n'étaient pas synchronisés avec le serveur et le plus gros problème était que si on déplaçait un joueur l'autre se déplacer également, enfaite on gérait tous les joueurs avec un seul et même joueur.

Nous avons donc réussi à synchroniser les intéractions des joueurs, c'est à dire que chaque instance voit bien l'autre se déplacer sur la plateforme. Pour cela, on a utilisé le component NetworkTransform qui permet de synchroniser chaque action des joueurs avec le serveur.

Pour régler le fait que un joueur contrôlait tous les joueurs, nous avons implémenté un script PlayerSetup qui vérifie si le joueur est bien notre joueur local à nous et va boucler sur tous les composants qui constitue le joueur (PlayerController qui gère les mouvements, camera...) et on va désactiver chaque composant de ce joueur. A partir de la si un joueur n'est pas le notre on n'aura plus le contrôle sur lui.

```
if (!isLocalPlayer)
{

    for (int i = 0; i < componentsToDisable.Length; i++)
    {
        componentsToDisable[i].enabled = false;
    }
```

Extrait du script PlayerSetup

Tout de même, nous avons rencontrer quelques problèmes comme le fait que lorsqu'une des deux instances se déplace vers l'autre et rentre en collision avec, l'instance touché se met à voler dans les airs. Ce problème sera le prochain objectif à résoudre sur le court terme.

Nos prochains objectifs pour le multijoueur vont être de bien changer les intéractions entre les joueurs comme les collisions par exemple.De plus, nous pensons que nous avons respecté nos attentes pour le multijoueur à ce stade (35%)

2.4 Intelligence Artificielle

2.4.1 Fonctionnement

Notre IA s'agit d'une entité qui erre dans la maison et qui chasse les joueurs si elle passe à proximité. Pour rendre le gameplay plus intéressant, j'ai aussi décidé de donner trois phases à cette IA.

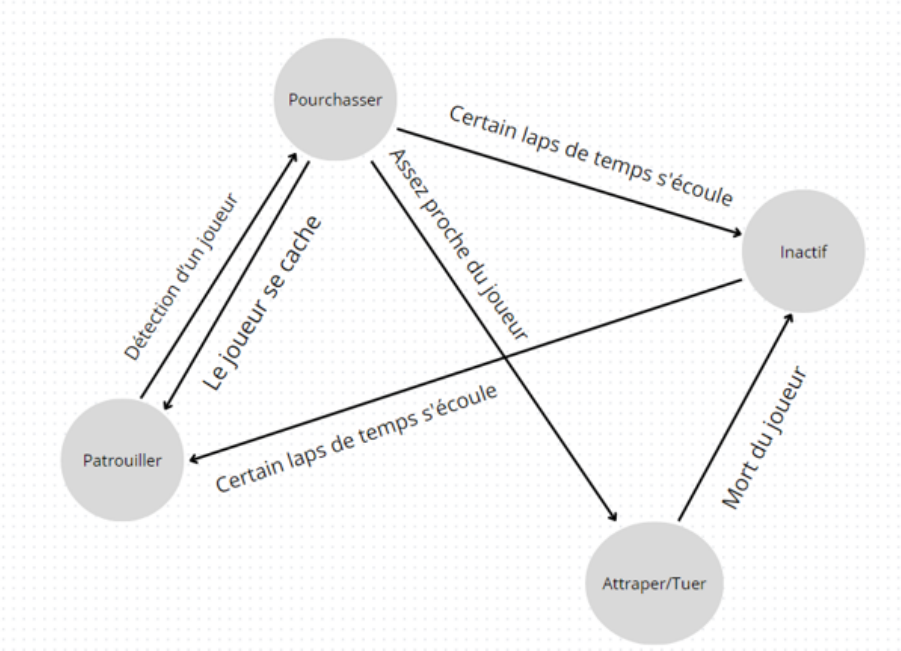
Tout d'abord, une première phase où l'IA est plutôt calme, apparaît relativement peu, et possède une petite zone de détection des joueurs. Ensuite, une seconde phase où l'IA est plus présente et où la zone de détection est élargie. Enfin, une troisième et dernière phase où l'IA est constamment présente, plus rapide, et dispose d'une zone encore plus grande.

Nous avons découvert qu'une IA FSM (« Finite State Machine ») serait la plus adaptée. Une IA FSM, ou machine à états finis en français, est notamment utilisée en robotique et dans les jeux vidéo. Elle repose sur trois caractéristiques principales : Les états : Ils représentent les différentes conditions ou situations dans lesquelles le système peut se trouver. Les transitions : Ce sont les règles qui dictent comment passer d'un état à un autre en réponse à des entrées ou des événements. Les entrées : Ce sont les données ou événements qui influencent les transitions.
Pour notre IA, les états pourraient être :

Être en patrouille : L'état où l'IA erre simplement dans la maison.

Être en chasse : L'état où l'IA poursuit un joueur.

La transition entre ces deux états pourrait être déclenchée par la détection d'un joueur. Enfin, l'entrée serait simplement l'événement de détection d'un joueur. L'avantage de ce type d'IA est qu'il est très lisible et facilement représentable à l'aide d'un graphique. De plus, il est très simple d'ajouter un état ou une transition entre différents états. Cependant, un inconvénient est que cette IA nécessite des connaissances solides en programmation orientée objet. Heureusement, grâce à l'excellente formation en programmation orientée objet de Jean-Malo, cela ne devrait pas poser de problème.



Voici un graphique représentant les états de base de notre IA (ce graphique est encore incomplet et sert simplement de base pour les éléments prioritaires de l'IA).

2.4.2 Algorithme

Maintenant que nous avons défini notre IA grâce à la FSM, je dois commencer la programmation. Pour cela, nous avons décidé de commencer par un algorithme essentiel au fonctionnement de cette IA : le pathfinding.

Le pathfinding, ou algorithme de recherche de chemin en français, est un algorithme qui, comme son nom l'indique, sert à trouver un chemin (idéalement le plus court) entre un point A et un point B. Nous avons choisi d'utiliser un type d'algorithme de pathfinding appelé A*. Cet algorithme est connu pour sa rapidité et son efficacité.

Cet algorithme fonctionne sur une grille. Sur cette grille, il y a :

-Une case de départ (appelée A).
-Une case d'arrivée (appelée B).
-Deux types de cases : les cases « empruntables » et les cases « non empruntables ».

Le fonctionnement de l'algorithme est le suivant. On définit un coût de déplacement d'une case à une autre : Par convention, ce coût est de 10 pour les déplacements en ligne ou en colonne, et de 14 pour les déplacements en diagonale. À partir de la case de départ, on calcule trois valeurs pour toutes les cases adjacentes :

-Coût G : La distance entre la case actuelle et la case de départ.

-Coût H : La distance entre la case actuelle et la case d'arrivée.

-Coût F : La somme de G et H (F = G + H).

Les cases où les valeurs sont calculées deviennent des « cases ouvertes ». L'algorithme choisit ensuite la case avec le coût F le plus faible parmi les cases ouvertes. En cas d'égalité, il choisit la case avec le coût H le plus faible. On répète le processus pour les cases adjacentes jusqu'à atteindre la case d'arrivée. Lorsque la case d'arrivée est atteinte, l'algorithme retrace le chemin emprunté depuis la case d'arrivée jusqu'à la case de départ pour déterminer le chemin le plus court

```
OPEN //the set of nodes to be evaluated
CLOSED //the set of nodes already evaluated
add the start node to OPEN

loop
 current = node in OPEN with the lowest f_cost
 remove current from OPEN
 add current to CLOSED

 if current is the target node //path has been found
  return

 foreach neighbour of the current node
  if neighbour is not traversable or neighbour is in CLOSED
   skip to the next neighbour

  if new path to neighbour is shorter OR neighbour is not in OPEN
   set f_cost of neighbour
   set parent of neighbour to current
   if neighbour is not in OPEN
    add neighbour to OPEN
```

Voici un exemple de pseudo-code de cet algorithme

Maintenant que nous connaissons cet algorithme, il nous faut l'implémenter. Pour ce faire, nous avons suivi le tutoriel de la chaîne YouTube Sebastian Lague, qui a réalisé une série de vidéos expliquant comment implémenter cet algorithme, mais aussi comment aller plus loin en ajoutant différentes améliorations à l'algorithme de base pour le rendre plus performant et lui donner plus de fonctionnalités.

Les étapes de ce tutoriel étaient tout d'abord d'implémenter une classe Node et un script Grid permettant de créer une grille à partir de la carte du jeu en identifiant les cases empruntables et non empruntables. Ensuite, nous avons implémenté le fameux algorithme cité précédemment. Puis, nous avons optimisé cet algorithme à l'aide d'un arbre de recherche.Par la suite, nous avons créé un script PathRequestManager permettant à plusieurs units de demander des chemins d'un point à un autre simultanément, sans provoquer de ralentissements.

Nous avons ensuite ajouté un système de poids pour éviter que l'IA emprunte certains types de terrain ou longe systématiquement les obstacles. Enfin, nous avons mis en place deux autres scripts : Line et PathF, permettant de donner une trajectoire plus naturelle à notre IA.

Finalement, je considère être légèrement en retard sur le planning de développement de notre IA. Cela est dû au fait que cette IA m'a demandé beaucoup de recherches. De plus, implémenter cet algorithme de pathfinding a été relativement compliqué, car il fait appel à des techniques de programmation plus avancées que celles que j'avais utilisées jusqu'à présent. J'ai également découvert trop tard que Unity possède un système de pathfinding intégré appelé NavMesh, mais je ne regrette pas complètement de l'avoir créé moi-même, car cela m'a permis de mieux comprendre comment fonctionne le pathfinding. Cette compréhension pourrait s'avérer très utile si je souhaite modifier ce code pour lui ajouter les fonctionnalités que je désire. Cela m'offre donc une plus grande liberté dans les ajouts futurs.

2.5 Décors et assets joueurs

2.5.1 Maison

Ce modèle représente un lit simple et sale, parfait pour créer une ambiance désordonnée et abandonnée dans les chambres.

Un lit double avec des draps déchirés et des matelas tachés, ajoutant un élément de réalisme sombre aux scènes d'intérieur.

Ce modèle plus ancien et plus dégradé suggère une longue période de négligence, idéal pour les scènes de flashback ou de souvenirs hantés.

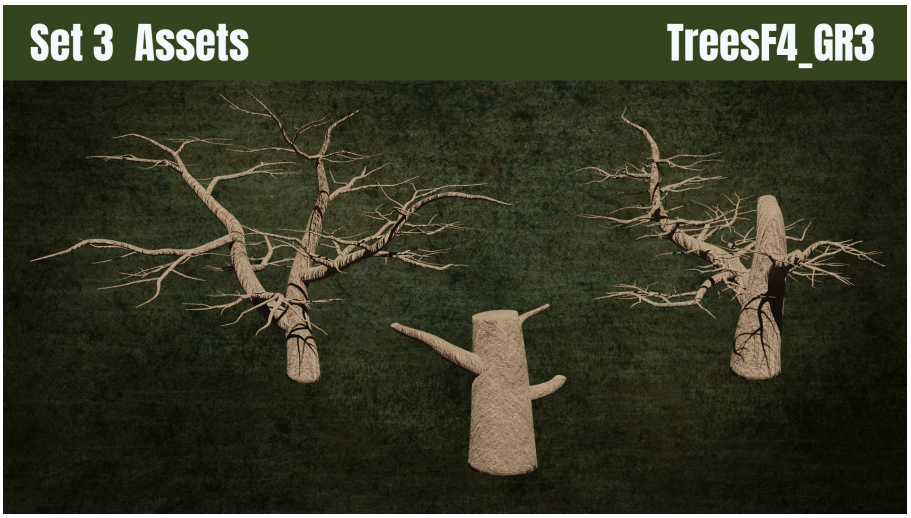
Un lit double avec des draps déchirés et des matelas tachés, ajoutant un élément de réalisme sombre aux scènes d'intérieur.

2.5.2 Design des joueurs

L'implémentation des assets dans le jeu n'a pas encore débuté, mais tout est prêt pour qu'elle commence très prochainement. Nous avons surtout rechercher les types d'assets que nous trouvons intéressantes pour notre jeu. Nous sommes donc à deux doigts de commencer l'intégration des assets dans notre jeu, avec l'objectif de garantir une immersion maximale pour les joueurs.

2.5.3 Décors extérieurs

Nous souhaitons également bien designer l'extérieur, tout autour de la maison, et pour rester dans notre thème on a décidé de partir sur une foret qui englobera la maison donc c'est pourquoi nous avons cherché et trouvé des assets correspondant.



Exemple d'asset d'arbres pour l'extérieur

2.6 Conception des missions

Depuis le début du projet, nous avons beaucoup réfléchi sur la conception des missions pour notre jeu d'horreur. Notre objectif est de rendre ces missions immersives, engageantes et techniquement solides pour offrir une expérience captivante aux joueurs.

2.6.1 Structure des missions

Jusqu'ici, nous n'avons pas encore implémenté de missions dans Unity, cependant nous avons une idée précise du type de mission et des exemples concrets rédigés. Nous avons réfléchi au fait que chaque mission correspondrait à une pièce : par exemple, dans la chambre, on aurait une mission associée à une poupée pour rester dans le thème de l'horreur. Ces missions doivent contribuer à l'ambiance et à la narration ; nous avons donc exploré différents types de missions :

- Résolution de missions simples : Rassembler 3 fragments d'un miroir maudit
- Interaction avec l'environnement : Allumer les bougies dans le bon ordre

Exemple de missions :

Pièce : Bibliothèque

- L'objectif est de retrouver un livre précis et le déposer sur le socle qui se trouve sur une petite table à l'entrée de la pièce. Ce livre est une description complète de l'entité présente dans la maison.

Pièce : Cuisine

- Le joueur doit trouver 3 fragments de papier pour reconstituer une vieille recette permettant de fabriquer une potion.

Actuellement nous avons tous les scripts rédigés de nos missions, nous savons exactement le déroulé de chacune d'entre elles.

2.6.2 Intégration de l'IA

L'intégration de l'IA joue un rôle central dans la dynamique de tension de notre jeu. Nous avons étudié comment utiliser le pathfinding, qui permet à l'entité hostile de naviguer dans l'environnement tout en évitant les obstacles. Une idée est d'associer l'IA aux missions : par exemple, si le joueur accomplit une mission ou fait du bruit en ramassant un objet, cela peut attirer l'attention de l'entité. Cela ajoute une couche de stress, car le joueur doit résoudre les missions rapidement tout en évitant d'être capturé.

2.7 Logo

Le logo de Lurking Home est conçu pour refléter l'essence même du jeu : mystérieux et effrayant.

Le logo est circulaire avec une typographie évoquant l'horreur. Des lignes irrégulières et des effets d'ombre créent une sensation d'inquiétude et de suspense. La forme circulaire symbolise l'inévitabilité et l'enfermement.

Une palette de couleurs sombres est utilisée pour renforcer l'aspect menaçant. Le noir symbolise l'inconnu, tandis que le rouge évoque le danger et le sang.

Une maison abandonnée, entourée de brouillard, est mise en avant. Le brouillard ajoute un élément de mystère, dissimulant ce qui pourrait se cacher au loin. Des éléments visuels comme des arbres dénudés et des ombres suggestives renforcent l'atmosphère de mystère.

Le titre du jeu est placé stratégiquement pour être visible tout en complémentant l'image de fond. Les éléments sont disposés de manière à guider le regard du spectateur vers le centre de l'image, où le mystère réside.

2.8 Bande sonore

2.8.1 Musique d'ambiance

La bande sonore est un élément fondamental pour immerger les joueurs dans l'univers de Lurking Home. Actuellement en phase de préparation pour être intégrée au jeu, elle promet de créer une tension constante, renforçant ainsi l'expérience d'horreur. Conçue pour évoluer en fonction des actions du joueur, la bande-son intensifie les moments critiques, rendant l'expérience encore plus immersive.

Des morceaux préliminaires ont été sélectionnés pour leur capacité à induire une sensation de malaise et de suspense. Chaque piste est testée pour s'assurer qu'elle réagit dynamiquement aux événements du jeu, créant une expérience audio réactive et engageante. Ces morceaux ont été choisis en fonction de l'ambiance que nous souhaitons instaurer dans les différentes pièces de la maison.

Nous avons également découvert une collection de musiques d'ambiance qui offrent des sons sinistres et des mélodies déstabilisantes, parfaites pour accompagner les scènes de tension et d'exploration. Les notes discordantes et les bruits de fond subtils contribuent à maintenir une tension constante, plongeant le joueur dans un état de vigilance permanent. Chaque morceau est conçu pour évoquer des émotions spécifiques, allant de l'anxiété à la terreur pure, afin de guider l'état émotionnel du joueur tout au long de l'histoire.

Pour accentuer le sentiment d'inconfort et d'inconnu, nous avons intégré des éléments de musique atonale. Des sessions d'écoute ont été organisées pour déterminer et sélectionner les sons les plus adaptés à notre vision artistique.

2.8.2 Effets sonores

En parallèle, des effets sonores tels que des craquements de plancher, des souffles de vent et des chuchotements discrets ajouteront une couche supplémentaire de réalisme et de frisson. Nous considérons que les effets sonores sont un aspect clé de l'ambiance dans les jeux d'horreur, ce qui explique l'attention particulière que nous leur avons portée.

Chaque bruit sera soigneusement synchronisé avec les événements du jeu afin de maximiser son impact sur le joueur. Ces effets sonores sont conçus pour être dynamiques, réagissant aux actions du joueur et ajoutant une dimension interactive à l'expérience auditive. Par exemple, le son des pas du joueur changera en fonction du type de sol, tandis que les bruits de fond varieront en fonction de l'environnement immédiat.

Enfin, nous souhaitons expérimenter avec la spatialisation du son afin de créer une sensation de direction et de distance, renforçant encore davantage l'immersion du joueur dans l'univers de Lurking Home.

2.9 Site web

Le site web est conçu pour directement mettre dans l'ambiance le joueur avant même qu'il lance le jeu. Il n'est pas notre priorité pour le moment, c'est pour cela qu'il n'y a que la structure de base du site d'implémenter.

2.9.1 Structure du site

La première étape a consisté à concevoir une structure HTML claire et adaptée à une navigation intuitive. Voici les principales réalisations dans ce domaine :

• En-tête (<header>) :
o Intégration d'un logo cliquable redirigeant vers la page d'accueil.
o Création d'un menu de navigation avec quatre sections : "Accueil", "Équipe", "Télécharger", et "Ressources". Les liens sont pour l'instant placeholders mais serviront à naviguer vers des pages ou sections spécifiques.

• Section principale (<main>) :
o Mise en place de deux blocs distincts pour structurer les informations clés :

1. Histoire du jeu : Description immersive soutenue par une image évoquant l'atmosphère oppressante du jeu.
2. Concept du gameplay : Présentation des mécanismes du jeu, accompagnée d'un visuel conceptuel.

2.9.2 Mise en page et esthétique

Pour refléter l'ambiance du jeu, une attention particulière a été portée au design visuel via une feuille de style CSS.

• Typographies :
o Pour les titres, nous avons temporairement utilisé la police Melted-Monster, qui donne un style effrayant en adéquation avec le thème du jeu.
o Les paragraphes sont stylisés avec Poppins-Black, une police moderne et lisible.
o Ces polices sont des propositions provisoires et pourront évoluer en fonction des tests utilisateurs.

• Palette de couleurs :

o Fond noir pour une ambiance sombre.
o Texte blanc pour un contraste net.

• Animation :

o Ligne rouge en dessous des liens apparait au survol de la souris

• Disposition :

o La mise en page utilise CSS Grid et Flexbox, qui permettent de structurer les éléments en grille pour une organisation claire et réactive.

Aperçu du site web

3 Conclusion

3.1 Rappels sur les avancées

Concernant l'histoire de notre jeu, elle est assez enrichi et possède des détails intéressants qui nous permettent de bien voir l'atmosphère de celui ci et l'idée global qu'on veut donner au jeu.

Lors de cette première soutenance, les différents points réalisés sont : une base multijoueur en local développée à l'aide de Mirror.

Nous avons également travaillé sur le gameplay, notamment sur la création des mouvements de base, des mouvements de la caméra ainsi que de la gestion de la gravité.

Nous avons aussi avancé sur l'IA avec une base de type FSM (Finite State Machine) et terminé un algorithme de pathfinding fonctionnel.

De plus, nous avons commencé à réfléchir aux différentes missions que notre jeu pourrait contenir. Nous avons également travaillé sur la structure de base du site internet.

Enfin, nous avons travaillé sur la direction artistique (DA) et les recherches d'assets, qu'ils soient visuels ou sonores.

Nous considérons que nous avons respecté tous nos objectifs sauf l'IA qui est tout juste et la recherche de design et décors.

3.2 Retards

Concernant les retards, nous avons remarqué que l'IA restait tout de même une tâche complexe donc nous considérons que nous avons un léger retard à ce niveau, nous espérions atteindre 25% mais nous n'y sommes pas encore ou alors tout juste.

La recherche de design est à poursuivre même si nous avons deja certaines assets pour la maison et pour l'extérieur, les assets des joueurs ne sont pas encore trouvés.

3.3 Les choses à faire pour la prochaine fois

Pour l'histoire, nous voulons l'améliorer et la terminer en rajoutant des indices sur l'histoire dans le jeu

Les points à améliorer pour la prochaine soutenance sont : mieux gérer les différentes interactions entre les joueurs et commencer les recherches pour héberger le multijoueur sur un serveur, afin de permettre la connexion et le jeu en ligne.

Nous aimerions également implémenter un système de ramassage d'objets, ainsi qu'un système de course avec une barre d'endurance.

De plus nous voudrions ajouter un moyen permettant à l'IA de détecter si un joueur passe à proximité, afin qu'elle puisse les poursuivre. Nous visons également une meilleure compréhension de l'implémentation d'une IA FSM dans Unity.

L'objectif est que, d'ici la prochaine soutenance, toutes les missions soient définies et que certaines commencent à être implémentées.

Nous aimerions créer les autres pages du site (Equipe, Ressources) et de commencer la chronologie de notre avancée.

Enfin, il faudrait qu'on possède plusieurs assets prêts à être implémentés dans notre jeu, avec une direction artistique claire à respecter.