



Rapport de projet

« Les maisons aux âmes perdues »

BERNARDEAU Lucas
JULIA Guillaume
BOCAGE Célia
BRAULT Valentin
VANHAESEBROUCKE Marius

Table des matières

1	Introduction	3
2	L'entreprise	4
2.1	Unununium	4
2.2	Les membres de l'équipe	5
3	Notre projet	7
3.1	Origine	7
3.2	Le scénario	8
3.3	Le but du jeu	8
3.4	Une expérience utile	8
3.5	Nos inspirations	9
3.5.1	Histoire	9
3.5.2	Gameplay	9
3.5.3	Graphisme	9
4	Synthèse de l'avancement du projet	10
4.1	Le Gameplay	10
4.2	Le Game design	17
4.3	Les Graphismes	26
4.4	Les menus	27
4.5	L'Intelligence Artificielle	29
4.6	Le Multijoueur & Réseau	38
4.7	Le Son	41
4.8	Le Site Web	41
5	Les problèmes rencontrés & les solutions apportées	44
5.1	Le Gameplay	44
5.2	Le Game design	48
5.3	Les Graphismes	49
5.4	Les Menus	49
5.5	L'Intelligence Artificielle	50
5.6	Le Multijoueur & Réseau	54
5.7	Le Son	54
5.8	Le Site Web	55
6	Points en suspens et réflexions pour une version ultérieure	56
6.1	Le Gameplay	56
6.2	Le Game design	57
6.3	Les Graphismes	57
6.4	Les Menus	58
6.5	L'Intelligence Artificielle	59
6.6	Le Multijoueur & Réseau	60
6.7	Le Son	60
6.8	Le Site Web	61

7 Réflexion personnelle sur l'expérience du projet	63
7.1 Lucas Bernardeau	63
7.2 Valentin BRAULT	64
7.3 Marius VANHAESEBROUCKE	65
7.4 Célia BOCAGE	65
7.5 Guillaume JUILA	66
8 Conclusion	67
9 Annexes	68
9.1 Annexe 1 : image illustrant la création de l'interface dans Godot	68
9.2 Annexe 2 : image illustrant l'interface utilisateur du menu de contrôle	68
9.3 Annexe 3 : Illustration de l'état "Attente de modification de touche"	69
9.4 Annexe 4 : Bougie éteinte	69
9.5 Annexe 5 : Bougie allumée	70
9.6 Annexe 6 : Une des poupées	70
9.7 Annexe 7 : Un des jouets	71
9.8 Annexe 8 : Menu victoire	71
9.9 Annexe 9 : Menu pause	72
9.10 Annexe 10 : Screamer du point de vue extérieur	72
9.11 Annexe 11 : Screamer du point de vue du joueur attrapé	72
9.12 Annexe 12 : Nœud AudioStremPlayer dans l'arborescence d'une scène	73
9.13 Annexe 13 : Paramètre du nœud AudioStremPlayer	73
9.14 Annexe 14 : Nouveau logo de l'entreprise	74
9.15 Annexe 15 : Icon de notre jeu vidéo	74
9.16 Annexe 16 : Image de notre page d'accueil	74
9.17 Annexe 17 : Image de notre page de téléchargement	75
9.18 Annexe 18 : QR code vers la page-cadeau	75
9.19 Annexe 19 : QR code vers notre site internet	75

1 Introduction

Ce rapport a pour objectif de vous présenter "Les Maisons aux Âmes Perdues", le premier opus d'une série de jeux vidéo prometteuse imaginée par le studio Unununium. Ce jeu d'aventure en coopération plonge les joueurs dans un univers cartoonesque empreint d'horreur et d'énigmes, promettant une expérience captivante en solo ou à deux.

Guidés par l'esprit d'un journaliste en quête de sensations fortes, les joueurs se retrouvent enfermés dans une maison hantée. Leur mission : s'échapper avant que le fantôme ne les capture définitivement. Pour y parvenir, ils devront résoudre une multitude d'énigmes disséminées à travers la maison, tout en échappant aux poursuites incessantes du fantôme.

"Les Maisons aux Âmes Perdues" se distingue par son gameplay riche et varié, mêlant savamment action, réflexion et coopération. Les joueurs devront non seulement user de leur agilité pour fuir le fantôme, mais également faire preuve de logique et d'esprit d'équipe pour résoudre les énigmes qui les attendent.

Pour une expérience encore plus saisissante, le jeu propose un environnement entièrement en 3D. Cette dimension immersive permet aux joueurs de se plonger pleinement dans l'atmosphère angoissante de la maison hantée et de vivre l'aventure avec une intensité redoublée.

Afin de renforcer l'atmosphère effrayante du jeu, "Les Maisons aux Âmes Perdues" s'accompagne d'une bande son et d'effets sonores soigneusement choisis. Des musiques angoissantes et des bruits saisisants rythmeront la progression des joueurs, contribuant à créer une expérience sensorielle inoubliable.

Ce rapport présente le premier niveau du jeu, servant de démonstration des mécaniques de jeu et de l'univers captivant des "Maisons aux Âmes Perdues". Cet aperçu prometteur laisse entrevoir le potentiel immense de cette série naissante, amenée à séduire les amateurs de jeux d'horreur et d'énigmes.

L'objectif principal de ce projet est de convaincre le jury de la qualité et du potentiel de "Les Maisons aux Âmes Perdues". Ce succès permettrait d'obtenir les ressources nécessaires pour développer une version complète du jeu, riche en contenu et offrant une expérience de jeu encore plus aboutie.

Fruit de l'imagination débordante et de la passion d'une équipe talentueuse, "Les Maisons aux Âmes Perdues" se veut être une expérience vidéoludique unique et mémorable. Ce projet ambitieux a pour but de marquer durablement le paysage des jeux d'horreur et d'énigmes, en proposant une aventure captivante, immersive et pleine de rebondissements.

2 L'entreprise

2.1 Unununium

Nous sommes une entreprise qui s'est formée le 27 septembre 2023 lors de notre sortie d'EPITA, soit trois mois après avoir été diplômés. Nous nous sommes regroupés autour des passions du jeu vidéo et de l'aventure. Tous passionnés par les énigmes, notre secteur d'activité principale est celui des jeux d'énigmes et d'aventures. Nous sommes une entreprise basée sur le partage et la joie. Toutes nos activités sont créées en ce sens. Nous organisons régulièrement des activités pour renforcer les liens qui nous unissent, comme des "escape game" ou des jeux grandeur nature, par exemple le jeu du "Loup Garou" ¹. C'est ainsi que l'idée de créer "Les maisons aux âmes perdues" est née, mais nous vous en parlerons plus précisément dans l'origine de notre projet.

Unununium peut paraître être un nom assez drôle, mais il a du sens pour nous. Lors de la création de nos premiers jeux, nous avons rencontré quelques problèmes dû aux systèmes d'exploitation (d'un point de vue des graphismes). En nous documentant, nous avons découvert que d'autre créateur avait été confronté au même problème. Néanmoins, certains se sont penchés sur le sujet en créant un système d'exploitation en temps réel graphique à composant.

L'équipe du projet décrivait ainsi ses objectifs : "Nous voulons un système complètement dynamique, où absolument aucun composant individuel n'est requis en mémoire de façon permanente, où la vitesse des applications en cours d'utilisation est de première importance, et où les programmeurs ont un total contrôle de la machine" ².

Ce projet était temporairement abandonné en mars 2007, pour reprendre en juin 2009. À nouveau abandonné en 2013 ³.

Place maintenant à la présentation de l'équipe en charge du projet.

1. Jeux de société : par Philippe des Pallières et Hervé Marly, ainsi que par les éditions Lui-même.

2. Source : Wikipédia

3. Source : Wikipédia & Dépôt forge (<https://sourceforge.net/projects/uuu/>)

2.2 Les membres de l'équipe

Lucas BERNARDEAU

- Poste : chef de projet et chargé du web et du Gameplay
- Surnom : Le président
- Devise : "La peur n'arrête pas le danger."⁴

Il est âgé de 18 ans. Il est passionné par l'informatique depuis le lycée. Il a commencé par coder de petits logiciels pour pirater son ancien ordinateur portal. Puis, il a découvert le monde du jeu vidéo. Cet autre univers de l'informatique lui a permis d'exprimer pleinement sa créativité. Il a commencé par créer des jeux comme "Le pendu", "Devine mon nombre entre 0 et 100" ou bien le "Tic-Tac-Toe" en python.

À sa sortie d'EPITA, il décide de créer une entreprise dans le jeu vidéo en recrutant ses anciens camarades de classes avec qui il entretient de bonnes relations. Ses capacités en programmation (C#, python, HTML/CSS, LaTex) lui permettent d'intervenir et de superviser le bon développement du projet.

Valentin BRAULT

- Poste : Chargé des menus et de L'Intelligence Artificielle
- Surnom : Nouille
- Devise : Rien n'est impossible.

Il est âgé 18 ans. Il est passionné d'informatique depuis son adolescence. Il découvre l'informatique dans un accompagnement en seconde et depuis il continue de coder des petits projets. En effet, il programme des mini-jeux du type "Snake" ou encore de type "morpion" en Python. C'est un passionné du jeu vidéo. Rejoindre Unununium était pour lui une évidence.

Cette entreprise, spécialisée dans divers types de jeux vidéo, lui correspondait parfaitement.

De plus, cette entreprise était dans son entourage une entreprise réputée tant pour ses conditions de travail que son ambiance. L'entreprise propose également des formations accompagnées et des activités d'intégration. En tant que jeune développeur, cet environnement de travail est un atout.

4. Eros Bonio : champion de France kickboxing et frère d'Enzo Bonio, deux fois champion du monde de kickboxing

Marius VANHAESEBROUCKE

- Poste : Chargé du multijoueur et du réseau
- Surnom : Buveur de café
- Devise : Nous sommes notre propre limite.

Depuis son plus jeune âge, il est passionné par les jeux vidéos. Il est curieux, il sait créer un jeu et surtout le rendre excellent et captivant.

Avant d'intégrer à l'EPITA, il faisait seul des petits projets comme des serveurs hébergés sur son ordi pour pouvoir jouer avec des amis. Il peut appliquer dans ce projet ses connaissances en C#, en Godot, en réseau ou encore en Blender. Ce projet est aussi pour lui l'occasion de faire son premier jeu. Il y a longtemps qu'il y songeait.

Célia BOCAGE

- Poste : Chargée du Gameplay et des Graphismes
- Surnom : Cana
- Devise : Force, courage et honneur.

Elle est une programmeuse diplômée de l'EPITA. Elle a grandi à la campagne avec ses deux frères également ingénieurs en informatique. Elle s'est rapidement intéressée à ce domaine et aux compétences qui s'en approchent. Adolescente, elle décide d'investir dans une tablette graphique pour apprendre à manipuler les logiciels de graphisme, autant pour du jeu vidéo ou même de l'animation.

Grâce à ses compétences graphiques. Après ses études, madame Bocage rejoint l'équipe Unununium en 2023. Elle fera en sorte d'aider du mieux possible ses anciens camarades d'EPITA, nouvellement collègues, afin de créer le meilleur jeu d'escape room possible.

Guillaume Julia

- Poste : Chargé du Gamedesign et de la musique
- Surnom : DL-Guigz-16
- Devise : "Hier est derrière, demain est mystère, et aujourd'hui est un cadeau, c'est pour cela qu'on l'appelle le présent."⁵

Il est passionné par l'horreur. Il adore les jeux vidéos. Il porte plus particulièrement son intérêt sur les jeux d'énigmes et d'investigations. Il participe souvent à des escape game. Grâce à son expérience dans les escape game, il s'occupera principalement du gameplay et participera aussi à la modélisation 3D.

À la sortie d'EPITA, il a choisi de rejoindre l'entreprise. Celle-ci lui correspond tant par l'ambiance de travail que par les grandes possibilités de création qu'offrent chaque projet.

5. Film : Kung Fu Panda

3 Notre projet

3.1 Origine

Notre équipe est adepte des « escape games » depuis un certain temps. C'est lors de notre première rencontre entre collègues que nous sommes tombés amoureux de ce genre d'activité. Nous nous connaissons à peine, notre week-end d'intégration avait pour but de faire connaissance et de créer les débuts de notre esprit de groupe, s'est organisé autour d'un « escape game ».

Mais qu'est-ce qu'un « escape game » ?

C'est un jeu collectif. Plusieurs personnes sont enfermées dans une ou plusieurs pièces dont elles doivent s'en échapper. Cette activité nous a permis de tisser des liens forts au sein de l'équipe qui y participe et c'était une vraie réussite pour nous. Ce jeu a permis de développer notre cohésion.

Cette première expérience nous a réellement marqués. Elle nous a plongé dans les conditions dans lesquelles la collaboration et l'entraide sont essentielles. Nous en gardons tous un excellent souvenir. Depuis, nous participons régulièrement à de nombreux « escape games » divers et variés. Nous améliorons notre sens logique et notre cohésion tout en passant un agréable moment. Cette activité nous a considérablement fait grandir aussi bien en tant que personnes qu'en tant qu'équipe.

Nous avions en commun, la recherche d'aventure. Depuis tout jeunes, certains membres de l'équipe étaient déjà très impliqués dans une pratique bien connue, mais rarement pratiquée : l'Urbex.

L'Urbex est une pratique qui consiste à visiter des lieux abandonnés difficiles d'accès comme d'anciens châteaux, d'anciennes demeures, etc. Cette pratique a pour but de jouer avec notre peur et de découvrir des lieux atypiques ou insolites.

Les membres qui la pratiquaient la présentèrent au reste du groupe. Nous nous sommes alors conciliés et l'idée nous plaisant, nous avons décidé de l'implémenter à notre projet. Nous l'avons donc exploitée afin de définir l'ambiance du jeu.

C'est pourquoi aujourd'hui notre équipe a décidé de créer un jeu dont le gameplay s'inspire directement de ces deux d'activités.

Notre jeu est un jeu de type action/aventure et de réflexion.

Pour ce qui est de la partie réflexion, il vous faudra collaborer avec votre partenaire, comme je l'ai évoqué précédemment, et cela, à travers diverses énigmes variées. Il vous faudra, en tant que joueur, réfléchir de votre côté et mettre en commun vos éléments avec ceux de votre partenaire dans l'objectif de vous échapper. Cette partie du jeu fait directement écho aux « escape games ».

Pour ce qui est de la partie aventure, puisque vous devez vous échapper, vous devrez résoudre des énigmes ! Il sera indispensable d'explorer minutieusement chaque pièce.

Pour ce qui est de la partie action, en plus de collaborer avec l'autre joueur, vous devrez éviter le fantôme qui circule dans la maison. Ainsi, cela vous demandera de vous cacher, d'être discret et parfois de courir.

3.2 Le scénario

Le scénario de “ Les maisons aux âmes perdues” est basé sur les enquêtes d'un journaliste que vous incarnerez.

Dans le cadre d'une revue sur le monde du mystique, celui-ci enquête sur des disparitions ou morts étranges ayant eu lieu dans différentes maisons du monde entier. Malheureusement, lors de ses enquêtes, une force mystérieuse pousse notre protagoniste à se retrouver enfermé dans les lieux qu'il inspecte. Lorsque le reporter s'en rend compte, cette même force lui indique sa mission : résoudre des énigmes qui lui permettront d'enquêter sur le fantôme du porté disparu et de s'échapper du piège dans lequel il s'est dirigé.

Le jeu semble alors être une tâche simple. Mais ne vous leurrez pas. Le fantôme de la victime ne reculera devant rien pour vous empêcher de découvrir le secret qui l'entoure. Vous allez donc, pour le bien de vos recherches, devoir l'éviter. Gare à vous, si l'entité démoniaque vous attrape ! Vous ne pourrez qu'en être affecté en mal. Dans la partie multijoueurs, le journaliste n'est pas seul, il est accompagné par un acolyte. Vous serez donc en équipe face au fantôme.

Dans le cadre de la démonstration de notre jeu, nous vous présenterons le premier niveau de celui-ci.

Il s'agit de l'histoire de Casper, un enfant mort seul dans une grande demeure en Angleterre... Serez-vous aptes à découvrir le secret qui l'entoure ?

3.3 Le but du jeu

Le but du jeu est de s'échapper seul ou à deux d'une maison abandonnée où circule un fantôme qui essaie de vous attraper. Il vous faudra collaborer avec votre partenaire afin que vous puissiez avancer ensemble. Si le fantôme vous attrape, votre champ de vision est réduit et vous réapparaissez dans un lieu d'apparition défini. Si votre écran devient totalement flou, vous mourez et c'est le « GAME OVER ». Finalement, vous aurez le choix. Si vous réussissez à vous échapper, vous libérerez ou pas l'âme du fantôme.

3.4 Une expérience utile

Nous avons créé ce projet pour gagner en expérience et en développement. Nous voulons également nous insérer dans l'industrie du jeu vidéo et accroître notre entreprise. Nous sommes tous passionnés par ce domaine et donc créer un jeu vidéo n'est que de l'enthousiasme pour notre équipe. Nous connaissons par ailleurs bien les rouages de ce milieu : ses concurrents, les principaux acteurs, les enjeux majeurs, etc.

Individuellement, ce projet permettra de développer nos compétences en programmation, de gagner en logique, en rigueur et en autonomie. Il faudra réfléchir à tous les cas possibles dans le game-design, être rigoureux dans notre code et notre capacité à nous adapter. Dans le cadre de ce projet complexe se déroulant en équipe. Notre créativité va se développer tout le long du projet, avec le choix des musiques, des textures, de l'intelligence artificielle et la façon dont seront construits les niveaux.

Ce projet apportera au groupe : un meilleur esprit d'équipe, mais aussi une meilleure ambiance. Le projet nous permettra de mieux nous connaître et d'apprendre à mieux communiquer. Il sera essentiel de bien communiquer puisque chacun devra travailler de son côté avant de tout regrouper. Et les tâches devront être correctement réparties, cela nous demandera d'être organisé.

3.5 Nos inspirations

Notre projet se déroule dans une demeure hantée par un fantôme orchestrant des énigmes dans un style rétro, et profondément influencé par divers médias.

3.5.1 Histoire

Du côté du cinéma, des œuvres telles que "Conjuring"⁶, "Dans le noir"⁷ et "Hérédité"⁸ ont été des influences majeures pour notre scénario. Ces films ont tous en commun la présence d'une entité malveillante traquant les protagonistes, un élément clé de notre histoire de jeu, créant ainsi une atmosphère d'horreur et de thriller immersive.

3.5.2 Gameplay

En ce qui concerne le côté réflexion et escape game, nous avons puisé des idées dans la série de jeux "Portal"⁹ et "Twelve Minutes"¹⁰, connus pour leurs énigmes complexes et leur gameplay captivant. Ces influences enrichissent l'aspect de résolution de casse-tête de notre jeu.

3.5.3 Graphisme

Sur le plan technique et graphique, nous nous sommes inspirés de jeux vidéos tels que "Phasmophobia"¹¹ et "The Mortuary Assistant"¹² pour capturer l'essence du thriller et de l'horreur. Ces jeux arrivent à créer une ambiance qui donne des frissons.

-
6. Conjuring : Film d'horreur d'un couple qui suit des événements paranormaux dans une maison hantée
 7. Dans le noir : Film d'horreur où les héroïnes sont hantées par une entité qui vit dans le noir
 8. Hérédité : Film avec des rituels sataniques et avec une maison hantée et des entités malveillantes
 9. Portal : Jeu d'énigme et de réflexion avec des mécaniques complexes
 10. Twelve Minutes : Jeu d'escape game avec une histoire technique
 11. Phasmophobia : Jeu vidéo d'horreur, il s'agit d'une chasse aux fantômes en coopération dans une maison
 12. The Mortuary Assistant : Un employé de la morgue se retrouve enfermé sur son lieu de travail, il s'y passe des événements paranormaux
-

4 Synthèse de l'avancement du projet

4.1 Le Gameplay

Tout d'abord, pour vous rappeler ce qui a été effectué précédemment à la dernière soutenance. Nous avions concentré les tests sur un environnement Windows et avons adapté le jeu pour les claviers AZERTY. De plus, nous avions programmé notre joueur en lui ajoutant les déplacements essentiels, le saut et la possibilité de s'accroupir. Nous avions ajouté la possibilité d'utiliser une lampe et la possibilité de ramasser des objets. Nous avions également ajouté des médicaments afin que le joueur puisse retrouver des points de vie s'il était attaqué par le fantôme. De plus, pour le gameplay, nous avions modifié la forêt qui entoure notre maison afin d'éviter les temps de chargement trop longs et les bugs possibles avec la génération aléatoire. Et nous avions rencontré de nombreux problèmes avec les escaliers.

Lors de la précédente présentation, nous avions défini quatre axes majeurs de développement : la compatibilité multi-plateforme, la personnalisation des touches, l'intégration d'animations 3D et l'optimisation des déplacements et des temps de chargement. Passons en revue les progrès réalisés sur chacun de ces points :

Afin de réduire les temps de latence et d'améliorer la fluidité des déplacements, nous avons procédé à une simplification des modèles 3D utilisés dans le jeu. Le nombre d'objets 3D a été réduit et la complexité polygonale de chaque objet a été optimisée. Cette optimisation s'est traduite par une diminution notable des temps de chargement et une expérience de jeu plus fluide.

Pour enrichir l'expérience visuelle et rendre les mouvements du personnage plus réalistes, nous avons intégré trois nouvelles animations 3D : une animation de course, une animation de marche et une animation "debout" pour maintenir le personnage immobile dans une posture droite. Ces animations ont été développées par l'équipe 3D et implémentées dans le code C# du jeu. Voici un extrait de code illustrant l'implémentation de l'animation de course et de la marche :

```
_animationPlayer.Play("standing");
//activation de l animation debout
//code inutile dans l explication
//
// code inutile dans l explication
//changement de vitesse si appuie sur bouton
if (Input.IsActionPressed("sprint") && (Input.IsActionPressed("forward") ||
    Input.IsActionPressed("back") || Input.IsActionPressed("right") ||
    Input.IsActionPressed("left")))
{
    MoveSpeed = RunningSpeed;
    _animationPlayer.Play("course");
    //activation de l animation de course
}
//code inutile dans l explication
//
// code inutile dans l explication
}
//touche de deplacement
if (Input.IsActionPressed("forward"))
{
    direction.Y -= 1;
    _animationPlayer.Play("Take 001");
    //activation de l animation de marche
    //le code est le mème pour les autres actions
}
```

L'activation de l'animation s'effectue en tirant parti des fonctionnalités natives de Godot. En effet, l'utilisation directe de la fonction PLAY permet de déclencher l'animation au sein du jeu.

Le jeu est désormais jouable sur Windows, Linux et Mac. Grâce à la puissance de Godot, la conversion du projet en un jeu exécutable sur ces différents systèmes d'exploitation s'est réalisée de manière fluide. Ces versions sont téléchargeables directement depuis notre site internet, sur la page dédiée (désormais fonctionnelle, comme nous le verrons par la suite).

La possibilité de personnaliser les touches constitue un aspect crucial de notre jeu. Pour concrétiser cette fonctionnalité, nous avons procédé comme suit :

1. La création de l'interface graphique : Un tableau affichant la liste des commandes modifiables a été créé.
2. L'ajout de boutons : Des boutons ont été intégrés à la liste afin d'établir un lien entre le programme de modification des touches et le jeu.
3. L'interface intuitive : L'interface obtenue présente une structure simple, avec la liste des actions à gauche et les touches associées à droite (Annexes 1 & 2).

Notre interface utilisateur présente un menu de contrôle cohérent et intuitif, en s'appuyant sur une police de caractère unique pour l'ensemble des éléments. Afin de vous familiariser avec son fonctionnement, nous allons vous détailler les principes clés de ce menu.

Un dictionnaire exhaustif a été créé pour répertorier toutes les touches pouvant être modifiées par l'utilisateur. Le code suivant illustre la mise en œuvre du dictionnaire et son interaction avec le menu de contrôle :

```
private Dictionary<string, string> input_actions = new  
Dictionary<string, string> {  
    {"forward", "Avancer"},  
    {"back", "Reculer"},  
    {"right", "Aller a droite"},  
    {"left", "Aller a gauche"},  
    {"space", "Sauter"},  
    {"accroupi", "S'accroupi"},  
    {"sprint", "Courir"},  
    {"lumiere", "Activer/Desactiver la Lumiere"},  
    {"regene", "Utiliser les medicaments"},  
    {"interact", "Interagir avec les objets"}}
```

Chaque action est associée à un texte descriptif qui s'affiche à l'écran. Lors de l'initialisation du menu, c'est-à-dire lors de sa création et de son apparition à l'écran, la fonction Ready() est appelée. Cette fonction récupère la liste des actions et appelle une autre fonction dédiée à la création d'une nouvelle liste d'actions.

```
public override void _Ready()  
{  
    actionList = GetNode(actionlist) as VBoxContainer;  
    _create_action_list();  
}
```

Le code source de la fonction _creation_action_list() est présenté ci-dessous. Une explication détaillée de cette fonction sera fournie dans la suite du document.

```
public void _create_action_list()
{
    InputMap.LoadFromProjectSettings();
    foreach (var item in actionList.GetChildren())
    {
        item.QueueFree();
    }
    foreach (var actios in input_actions.Keys)
    {
        Button button = (Button) ResourceLoader.Load<PackedScene>(inputbutton)
            .Instantiate<Node>();
        var action_label = (Label) button.FindChild("LabelAction");
        var input_label = (Label) button.FindChild("LabelInput");
        action_label.Text = input_actions[actios];
        var events = InputMap.ActionGetEvents(actios);
        if (events.Count > 0)
        {
            input_label.Text = events[0].AsText().TrimSuffix(" (Physical)");
        }
        else
        {
            input_label.Text = "";
        }
        actionList.AddChild(button);
        button.Pressed += () => _on_input_button_pressed(button, actios);
    }
}
```

La nature volumineuse de ce bloc de code rend sa compréhension immédiate difficile. Pour favoriser une approche plus claire et progressive, nous allons le décomposer en éléments plus petits et analyser chaque partie en détail. Nous commencerons par examiner les premières lignes et la première boucle.

```
InputMap.LoadFromProjectSettings();
foreach (var item in actionList.GetChildren())
{
    item.QueueFree();
}
```

L'élément central de notre processus est l'actionList, qui sert de conteneur pour toutes les actions à exécuter. La fonction InputMap.LoadFromProjectSettings() permet de charger les paramètres du projet, comme son nom l'indique. La boucle suivante vide l'actionList afin de la remplir avec nos propres actions personnalisées.

Passons maintenant à la boucle principale, qui constitue l'élément central du traitement. Cette boucle effectue l'essentiel du travail en itérant sur chaque élément de la liste d'actions et en exécutant l'action correspondante.

Ce code présente la structure de notre boucle principale :

```
foreach (var actios in input_actions.Keys)
{
    Button button = (Button) ResourceLoader.Load<PackedScene>
        (inputbutton).Instantiate<Node>();
    var action_label = (Label) button.FindChild("LabelAction");
    var input_label = (Label) button.FindChild("LabelInput");
    action_label.Text = input_actions[actios];
    var events = InputMap.ActionGetEvents(actios);
    if (events.Count > 0)
    {
        input_label.Text = events[0].AsText().TrimSuffix(" (Physical)");
    }
    else
    {
        input_label.Text = "";
    }
    //code decris par la suite
}
```

Cette boucle permet d'établir une correspondance entre chaque action du dictionnaire et sa touche correspondante. Pour chaque action, un nouveau bouton est créé, son nom affiché est modifié, et le nom de la touche associée est ajusté. Un test conditionnel ("if") permet d'associer la touche adéquate en fonction des conventions du projet, en supprimant la chaîne de caractères "(Physical)" uniquement à des fins d'esthétique du menu. Le nouveau bouton est ensuite ajouté au tableau et sa fonctionnalité de détection d'appui est vérifiée.

Le code associé :

```
actionList.AddChild(button);
button.Pressed += () => _on_input_button_pressed(button, actios);
```

Lors de l'appui sur le bouton, la fonction `_on_input_button_pressed` est déclenchée. Cette fonction prend en paramètre le bouton en question ainsi que le nom de l'action qui lui est associée.

Code associé à la fonction :

```
private void _on_input_button_pressed(Button button, string action)
{
    if (!is_remapping)
    {
        is_remapping = true;
        action_to_remap = action;
        remapping_button = button;
        Label t = (Label) button.FindChild("LabelInput");
        t.Text = "Appuyez sur une touche pour modifier";
    }
}
```

Cette fonction permet de déterminer si une touche est actuellement en cours de modification. Si aucune touche n'est modifiée, le texte "Appuyez sur une touche pour modifier" s'affiche à la place de la touche associée. Pour plus d'informations, veuillez consulter l'Annexe 3.

Grâce à ce code, la fonction Input de Godot prend en charge le remplacement direct de l'association d'une touche par une nouvelle, comme l'indique l'exemple ci-dessous.

```
public override void _Input(InputEvent ev)
{
    if (is_remapping)
    {
        if (ev is InputEventKey ||
            (ev is InputEventMouseButton && ev.IsPressed()))
        {
            if (ev is InputEventMouseButton e && e.DoubleClick)
            {
                e.DoubleClick = false;
            }
            InputMap.ActionEraseEvents(action_to_remap);
            InputMap.ActionAddEvent(action_to_remap, ev);
            _update_action_list(remapping_button, ev);
            is_remapping = false;
            action_to_remap = null;
            remapping_button = null;
            AcceptEvent();
        }
    }
}
```

Cette fonction remplace le double clic par un simple clic afin de prévenir les problèmes potentiels dans notre jeu. La liste des actions est ensuite mise à jour via la fonction update_action_list.

Le code associé :

```
public void _update_action_list(Button button, InputEvent ev)
{
    Label t = (Label) button.FindChild("LabelInput");
    t.Text = ev.AsText().TrimSuffix(" (Physical)");
}
```

Elle se charge de remplacer la touche associée par la nouvelle et mettre à jour notre tableau de touche.

Lors de la précédente soutenance, la question de la détection des interactions entre le joueur et les escaliers a été soulevée. Après analyse, il a été décidé de conserver la configuration actuelle basée sur le principe de "boîte" pour la détection des interactions.

Avantages de la solution actuelle :

- Simplicité de mise en œuvre
- Fonctionnalité des escaliers

Inconvénients de la solution actuelle :

- Saut involontaire du joueur à la descente, même sans appui sur la touche

Malgré le désagrément du saut involontaire, le choix a été fait de maintenir la solution actuelle pour des raisons de pragmatisme. En effet, cette approche garantit la fonctionnalité des escaliers, ce qui constitue un élément crucial pour la jouabilité du jeu.

4.2 Le Game design

Cette partie présente en détail la conception de notre jeu vidéo, mettant en scène un journaliste enquêtant sur un fantôme. Le joueur incarne ce journaliste et doit explorer un manoir hanté, interagir avec des objets et résoudre des énigmes pour progresser dans l'histoire, se libérer de la maison et obtenir des informations pour son article.

Les mécaniques de jeu :

- L'exploration et la découverte : Le joueur explore librement le manoir et ses environs, découvrant des indices et des éléments narratifs.
- La résolution d'énigmes : Des énigmes non linéaires, basées sur la logique et l'observation, permettent au joueur de progresser.
- Les interactions avec des objets : De nombreux objets, chacun avec des fonctions et des mécanismes uniques, enrichissent l'expérience de jeu.
- Le récit captivant : L'histoire se dévoile progressivement à travers des indices et des "notebooks", immergeant le joueur dans le mystère.

L'accent est mis sur une approche non linéaire, permettant aux joueurs de relever les défis dans l'ordre de leur choix. Cette liberté favorise l'exploration et l'immersion, chaque joueur progressant à son rythme. Bien que les énigmes soient liées, leur ordre de résolution n'est pas crucial, renforçant l'autonomie du joueur.

Divers objets disséminés dans l'environnement offrent des interactions uniques :
La classe bougie : Cette classe gère la mécanique de collecte et d'allumage des bougies, essentielles pour progresser dans certaines zones et révéler des indices.

Dans cet environnement, des bougies sont disséminées, invitant le joueur à les découvrir et les illuminer pour progresser. Chaque bougie éteinte trouve son pendant allumé, symbolisant une renaissance ou une résurgence de souvenirs enfouis. En interagissant avec une bougie éteinte, le joueur la rallume, dévoilant ainsi un fragment de l'histoire tout en contribuant à l'illumination progressive de l'environnement. L'allumage de l'ensemble des bougies déverrouille un objet spécial, marquant une étape charnière dans l'aventure du joueur.(Annexe 4 & 5)

Entrons maintenant dans le détail du code :

```
private List<Area3D> allObjects;
private bool Dedans;
private Area3D CurrentArea;
private int collectedObjects = 0;
private Area3D specialObject;
private Dictionary<Area3D, Area3D> candleMapping;

public override void _Ready()
{
    var bougieEteinte1 = GetNode<Area3D>("/root/Node3D/Bougie_eteinte");
    var bougieEteinte2 = GetNode<Area3D>("/root/Node3D/Bougie_eteinte2");
    var bougieEteinte3 = GetNode<Area3D>("/root/Node3D/Bougie_eteinte3");
    var bougieAllu1 = GetNode<Area3D>("/root/Node3D/BougieAllu");
    var bougieAllu2 = GetNode<Area3D>("/root/Node3D/BougieAllu2");
    var bougieAllu3 = GetNode<Area3D>("/root/Node3D/BougieAllu3");
    allObjects = new List<Area3D>
    {
        bougieEteinte1,
        bougieEteinte2,
        bougieEteinte3,
    };
    candleMapping = new Dictionary<Area3D, Area3D>
    {
        { bougieEteinte1, bougieAllu1 },
        { bougieEteinte2, bougieAllu2 },
        { bougieEteinte3, bougieAllu3 }
    };
    foreach (var candle in candleMapping.Values)
    {
        candle.Hide(); // Initially hide all lit candles
    }
    specialObject = GetNode<Area3D>("/root/Node3D/Notebook3");
    specialObject.Hide();
}
```

La fonction `_Ready()` est exécutée lorsque l'objet est prêt à être utilisé. Elle effectue les initialisations nécessaires, telles que :

- Création des listes d'objets : bougies éteintes et bougies allumées.
- Création d'un dictionnaire pour associer chaque bougie éteinte à sa version allumée.
- Masquage initial des bougies allumées et de l'objet spécial.

```
public override void _Process(double delta)
{
    if (Input.IsActionPressed("interact") && Dedans)
    {
        Synch();
        Rpc("Synch");
    }
}
```

Cette fonction est exécutée chaque itération de la boucle de jeu. Elle est responsable de la vérification de l'interaction du joueur avec une bougie et de la gestion de la proximité du joueur avec cette dernière. En cas d'interaction ou de proximité, la fonction synchronise les actions du joueur avec les autres joueurs participants via le réseau.

Fonctionnement détaillé :

- Vérification de l'interaction : La fonction détermine si le joueur est en train d'interagir avec une bougie. Cela peut impliquer la détection d'une pression sur un bouton, d'un mouvement du curseur ou de toute autre action définie comme une interaction avec une bougie.
- Vérification de la proximité : La fonction évalue la distance entre le joueur et la bougie. Si la distance est inférieure à un certain seuil prédéfini, le joueur est considéré comme étant à proximité de la bougie.
- Synchronisation réseau : Si une interaction ou une proximité est détectée, la fonction déclenche la synchronisation des actions du joueur avec les autres joueurs sur le réseau. Cela garantit que tous les joueurs voient la même situation et que les interactions avec la bougie sont cohérentes pour tous.

```
private void Synch()
{
    // Hide only the current area and show the corresponding lit candle
    if (CurrentArea != null)
    {
        CurrentArea.Hide();
        if (candleMapping.ContainsKey(CurrentArea))
        {
            candleMapping[CurrentArea].Show();
            // Show the corresponding lit candle
        }
        collectedObjects++; // Increment the collected objects count
        CheckCollectedObjects(); // Check if all objects are collected
    }
}
```

Cette fonction gère la synchronisation de la collecte et de l'allumage des bougies.

Fonctionnement :

- Masque la bougie éteinte actuelle : Rend la bougie éteinte actuellement sélectionnée invisible.
- Affiche la bougie allumée correspondante : Remplace la bougie éteinte masquée par sa version allumée.
- Incrémente le compteur d'objets collectés : Met à jour le décompte du nombre d'objets récupérés avec succès.
- Vérifie si tous les objets ont été collectés : Détermine si l'objectif de collecte a été atteint.

```
private void _on_body_entered(Node3D body)
{
    if (body is CharacterBody3D)
    {
        if (((player)body).Id == Multi_game_manager.Players[0].Id)
        {
            Dedans = true;
            CurrentArea = GetCurrentArea(body);
        }
    }
}
```

Cette fonction est déclenchée lorsque le joueur pénètre dans la zone d'influence d'une bougie. Elle vérifie si le joueur en question correspond au joueur local (celui contrôlé par l'utilisateur) et met à jour la zone actuelle dans laquelle se trouve ce dernier.

```
private void _on_body_exited(Node3D body)
{
    if (body is CharacterBody3D)
    {
        if (((player)body).Id == Multi_game_manager.Players[0].Id)
        {
            Dedans = false;
            CurrentArea = null; // Clear the current area when the player exits
        }
    }
}
```

Cette fonction est déclenchée lorsque le joueur contrôlé localement quitte la zone d'influence d'une bougie. Elle vérifie si le joueur concerné est bien celui contrôlé par le joueur local et, si c'est le cas, procède à la réinitialisation de la zone actuelle.

```
private Area3D GetCurrentArea(Node3D body)
{
    // Function to determine which area the player is currently in
    foreach (var area in allObjects)
    {
        if (area.GetOverlappingBodies().Contains(body))
        {
            return area;
        }
    }
    return null;
}
```

Cette fonction détermine la zone dans laquelle se trouve le joueur en identifiant les zones d'objets qui se superposent au joueur.

```
private void CheckCollectedObjects()
{
    // Check if the player has collected all objects
    if (collectedObjects >= allObjects.Count)
    {
        specialObject.Show(); // Show the special object
    }
}
```

La fonction CheckCollectedObjects permet de vérifier si tous les objets ont été collectés. Si l'ensemble des objets est effectivement collecté, la fonction affiche un objet spécial.

La classe Sœurs est responsable de la gestion de la collecte des poupées dans le jeu. Elle joue un rôle central dans l'enrichissement de l'histoire et du développement du personnage principal.

Fonctionnalités :

- Collecte de poupées : Les poupées dispersées dans l'environnement représentent des fragments précieux des souvenirs d'enfance du personnage principal.
- Déblocage de l'histoire : Chaque poupée收集ée agit comme une clé, déverrouillant des segments de l'histoire du personnage. Ces révélations éclairent son passé, les événements qui ont façonné sa vie et les circonstances qui l'ont mené à sa situation actuelle.
- Révélations cruciales : En rassemblant toutes les poupées, le joueur accède à des vérités essentielles sur l'identité et le destin du personnage principal, enrichissant ainsi considérablement sa compréhension de l'histoire.

Pour visualiser les images, reportez-vous à l'annexe 6.

Entrons maintenant dans le détail du code :

```
public override void _Ready()
{
    allObjects = new List<Area3D>
    {
        GetNode<Area3D>("/root/Node3D/Froggie"),
        GetNode<Area3D>("/root/Node3D/Foxie")
    };
    specialObject = GetNode<Area3D>("/root/Node3D/Notebook4");
    specialObject.Hide();
}
```

Cette fonction effectue les tâches suivantes :

- Initialisation de la liste des poupées : La fonction crée et initialise une liste vide pour stocker les poupées. Cette liste sera utilisée pour suivre les poupées présentes dans le jeu.
- Masquage de l'objet spécial : L'objet spécial est initialement caché pour éviter qu'il ne soit visible par le joueur avant le début du jeu. Cela permet de créer une sensation de mystère et d'intrigue.

La fonction `_Process` gère l'interaction du joueur avec les poupées. Elle détecte si le joueur se trouve à proximité d'une poupée et si une interaction est en cours. Si c'est le cas, elle synchronise l'action de collecte avec les autres joueurs via le réseau afin de garantir une expérience cohérente pour tous les participants.

La fonction `Synch` centralise la logique de collecte des poupées. Elle masque la poupée collectée pour les joueurs, met à jour le compteur d'objets collectés et vérifie si toutes les poupées ont été rassemblées. Si tel est le cas, elle déclenche l'affichage de l'objet spécial, qui peut être une récompense ou un élément important pour la progression du jeu.

Les fonctions `_on_body_entered` et `_on_body_exited` gèrent la détection des entrées et sorties des zones de poupées. Elles permettent de mettre à jour la zone actuelle du joueur en fonction de sa position dans l'environnement du jeu.

La fonction `GetCurrentArea` est un utilitaire qui détermine la zone actuelle du joueur en identifiant les zones d'objets qui chevauchent sa position.

Enfin, la fonction `CheckCollectedObjects` vérifie si tous les objets ont été collectés. Cette fonction est appelée périodiquement ou après chaque action de collecte pour surveiller la progression du joueur et déclencher les événements appropriés lorsque tous les objets ont été rassemblés.

En résumé, ces fonctions travaillent ensemble pour offrir une expérience de jeu fluide et interactive en ce qui concerne la collecte des poupées. Elles gèrent la détection des interactions, la synchronisation des actions entre les joueurs, la mise à jour des scores et l'affichage des récompenses.

La classe "Jouets" est responsable de la gestion de la collecte des objets de collection dispersés dans l'environnement de jeu. Ces objets, empreints de nostalgie et d'innocence, servent à retracer les moments heureux du personnage principal.

Pour visualiser les images, reportez-vous à l'annexe 7.

Initialisation du jeu :

_Ready() :

- Initialise la liste des jouets présents dans le jeu.
- Cache l'objet spécial dans un premier temps.

Gestion des interactions avec les jouets :

_Process(double delta) :

- Vérifie si le joueur interagit avec un jouet en entrant dans sa zone d'influence.
- Si une interaction est détectée :
 - Synchronise l'action de collecte avec les autres joueurs via le réseau.
 - Met à jour l'état du jeu en conséquence.

Synchronisation de la collecte des objets :

Synch() :

- Synchronise l'action de collecte d'un jouet entre les joueurs.
- Cache le jouet collecté pour tous les joueurs.
- Incrémente le compteur d'objets collectés.
- Vérifie si tous les objets ont été collectés.
- Si tous les objets ont été collectés, affiche l'objet spécial.

Gestion des zones d'objets :

_on_body_entered(Node3D body) :

- Met à jour la zone actuelle du joueur lorsqu'il entre dans la zone d'influence d'un jouet.

_on_body_exited(Node3D body) :

- Réinitialise la zone actuelle du joueur lorsqu'il quitte la zone d'influence d'un jouet.

GetCurrentArea(Node3D body) :

- Détermine la zone actuelle du joueur en identifiant la zone d'objet qui chevauche sa position.

Vérification de la fin du jeu :

CheckCollectedObjects() :

- Vérifie si tous les objets ont été collectés.
- Si tous les objets ont été collectés, affiche l'objet spécial et déclenche la fin du jeu.

La classe Kee1 gère l'interaction du joueur avec un élément crucial du jeu : une clé.

L'objet clé :

La clé est un objet unique et indispensable qui permet au joueur de débloquer une zone spéciale ou un élément de jeu essentiel. Elle symbolise la découverte d'une vérité cachée ou la résolution d'une énigme complexe, marquant un tournant dans l'aventure du joueur.

Fonctions de la classe Kee1 :

_Ready() : Initialise la variable représentant la clé.

_Process(double delta) :

- Vérifie si le joueur interagit avec la clé et s'il est à proximité.
- Si c'est le cas, synchronise l'action avec les autres joueurs via le réseau.

Synch() : Synchronise l'action d'interaction avec la clé en la masquant.

OnCollision(Node3D body) :

- Appelée lorsque le joueur entre en collision avec la zone de la clé.
- Vérifie si le joueur est celui contrôlé par le joueur local.
- Met à jour la variable indiquant que le joueur est à proximité de la clé.

OutCollision(Node3D body) :

- Appelée lorsque le joueur quitte la zone de la clé.
- Vérifie si le joueur est celui contrôlé par le joueur local.
- Met à jour la variable indiquant que le joueur n'est plus à proximité de la clé.

_on_kee_1_visibility_changed() :

- Déclenchée lorsque la visibilité de l'élément change.
- Réactive la collision de la porte associée à la clé pour permettre son ouverture après la collecte de la clé.

La classe Kee1 gère l'interaction avec un élément crucial du jeu qui permet la progression du joueur. Elle assure une synchronisation fluide de l'action d'interaction avec la clé entre les joueurs en réseau. La visibilité de la clé et la collision avec la porte associée sont gérées de manière cohérente.

Les Cahiers, allant de 0 à 7, jouent un rôle crucial dans l'aventure en servant à la fois d'indices pour déchiffrer des énigmes et d'éléments narratifs enrichissant l'histoire du fantôme.

Fonction d'indices :

- Guidant les joueurs vers des objets cachés et des énigmes intrigantes, les indices stimulent la curiosité et l'exploration.

Exemple :

- "Plongez dans l'univers des poupées russes, explorez chaque strate de leur mystère et laissez-vous emporter par l'histoire qu'elles renferment pour découvrir la vérité cachée en leur cœur."

Cet indice oriente les joueurs vers des poupées russes dissimulées, créant une atmosphère captivante qui les incite à les rechercher activement.

Fonction narrative :

- Les Cahiers révèlent des fragments de l'histoire du fantôme et de son passé, immergeant davantage les joueurs dans l'univers du jeu.

Exemple :

- "Parmi les artefacts spectraux, Froggie et Foxie se distinguent comme gardiens de souvenirs. Froggie, témoin silencieux des joies et des peines de l'enfance, évoque des moments perdus. Foxie, au regard perçant et au sourire énigmatique, incarne le mystère. Ensemble, ils sont les gardiens des secrets enfouis."

Cette description détaillée de Froggie et Foxie offre un aperçu du lien émotionnel du fantôme avec ces objets et enrichit le récit global.

Pour garantir une expérience fluide en multijoueur, des mécanismes de synchronisation et d'identification des personnages ont été mis en place :

Synchronisation des actions :

- Les actions des joueurs, comme ramasser des objets ou activer des éléments, sont synchronisées pour que tous les joueurs observent les événements simultanément, peu importe la latence ou la performance de leurs appareils.

Identification des personnages :

- Chaque joueur est associé à un identifiant unique qui permet d'attribuer correctement les actions et d'assurer que les événements correspondent à l'activité de chaque participant, même dans des environnements multijoueurs complexes.

En résumé, les Cahiers et les mécanismes multijoueurs contribuent à créer une expérience immersive et captivante, en stimulant la découverte, en enrichissant le récit et en garantissant une interaction fluide entre les joueurs.

4.3 Les Graphismes

Cette partie présente les modifications apportées à la carte du jeu suite au dernier rapport de soutenance. Ces modifications visent à améliorer la qualité de l'expérience de jeu en corigeant des problèmes de collision et en ajoutant un environnement extérieur.

Dans le dernier rapport de soutenance, il a été signalé que la hitbox générée par Godot pour le sol de la carte présentait des irrégularités. Cela entraînait des problèmes de collision où le personnage pouvait traverser le sol. La complexité de la géométrie du sol compliquait la génération de la hitbox et contribuait à ces problèmes.

Pour corriger ces problèmes, le sol a été simplifié et remodelé. Cette simplification a permis de générer une hitbox plus précise et d'éliminer les problèmes de collision. De plus, la performance du jeu a été améliorée grâce à la réduction de la complexité de la géométrie du sol.

Un environnement extérieur a été ajouté à la carte pour créer une expérience de jeu plus immersive et cohérente. Cet environnement comprend des éléments tels que des arbres, des rochers et de l'herbe.

Les premiers tests ont montré que l'ajout de l'environnement extérieur n'a pas entraîné de nouveaux problèmes de collision.

La conception de graphismes et de modèles 3D de qualité s'avérant complexe, nous avons fait appel à des banques de ressources externes pour enrichir notre environnement virtuel. Cette intégration d'éléments visuels provenant de sources externes nous a permis d'aménager la maison présente dans notre jeu avec un mobilier varié et d'intégrer des animations dynamiques. Ces ajouts contribuent à une immersion accrue des joueurs, renforçant le réalisme et la vitalité de l'expérience utilisateur.

Soucieux de l'harmonie visuelle de notre jeu, nous avons accordé une attention particulière à la sélection de la police d'écriture pour les menus. Notre choix s'est porté sur la police "Crunchy Time" pour l'ensemble des textes visibles, car son style correspond parfaitement à l'esprit et à l'ambiance que nous souhaitons véhiculer. Cette cohérence typographique participe à l'affirmation de l'identité visuelle du jeu, captivant les joueurs par son esthétique harmonieuse.

La combinaison de ressources graphiques externes et d'un choix typographique judicieux a permis d'optimiser l'expérience utilisateur au sein de notre jeu. L'environnement visuel enrichi et la cohérence esthétique contribuent à une immersion plus profonde des joueurs, rendant l'expérience de jeu plus captivante et agréable.

4.4 Les menus

L'implémentation de menus intuitifs et conviviaux est un aspect crucial de tout jeu vidéo, permettant aux joueurs de naviguer aisément et d'accéder aux différentes fonctionnalités du jeu. Lors de la dernière soutenance, notre équipe a présenté l'avancement significatif réalisé sur ce volet, surpassant l'objectif initial de 60% et finalisant ainsi 80% des menus du jeu.

Le menu principal accueille les joueurs dès le lancement du jeu et propose les options suivantes :

- Jouer seul : Démarrer une partie en mode solo.
- Multijoueur : Accéder aux options de jeu en ligne.
- Paramétrage des contrôles : Personnaliser les commandes du jeu selon les préférences du joueur.
- Quitter : Fermer le jeu.

Le menu multijoueur offre les fonctionnalités suivantes :

- Héberger : Créer une partie et inviter d'autres joueurs à la rejoindre.
- Adresse IP à rejoindre : Se connecter à une partie en cours en saisissant son adresse IP.
- Rejoindre : Se connecter à une partie existante hébergée par un autre joueur.
- Retour : Quitter le menu multijoueur et revenir au menu principal.

Pour lancer une partie multijoueur, les joueurs doivent tous appuyer sur le bouton "Prêt" puis sur "Démarrer". Ce menu gère également les erreurs de connexion de manière conviviale, informant clairement les joueurs des problèmes rencontrés.

Accessible en appuyant sur la touche "Echap" pendant le jeu, le menu pause propose les options suivantes :

- Continuer : Reprendre la partie en cours.
- Paramétrage des contrôles : Modifier les commandes du jeu sans quitter la partie.
- Quitter le jeu : Fermer le jeu et revenir au menu principal.

Ce menu s'affiche après la défaite du joueur et propose les options suivantes :

- Recommencer : Démarrer une nouvelle partie en conservant la progression.
- Quitter le jeu : Fermer le jeu et revenir au menu principal.



Lors de la dernière présentation, nous avons exprimé notre volonté d'enrichir l'expérience utilisateur de notre jeu en introduisant de nouvelles fonctionnalités via deux menus supplémentaires. Ces ajouts visent à offrir une expérience plus profonde et plus personnalisée, répondant ainsi aux attentes de notre communauté de joueurs.

Désormais, une douce symphonie de victoire retentit lorsque vous triompez des défis du jeu. Un menu de victoire s'affiche, offrant une conclusion gratifiante à votre aventure. Ce menu propose plusieurs options :

- Rejouer : Plongez immédiatement dans une nouvelle partie, reprenant votre quête avec ferveur.
- Retour au menu principal : Accédez à l'écran principal du jeu, vous permettant de faire une pause bien méritée.

Ce menu de victoire, s'inspirant de son homologue du menu "Game Over", a été conçu avec fluidité pour une expérience utilisateur homogène.(Annexe 8)

Affirmez votre style de jeu unique grâce au menu de personnalisation des contrôles. Configurez les commandes selon vos préférences, pour une expérience de jeu optimale et accessible. Cette fonctionnalité s'adresse à tous les joueurs, qu'ils affectionnent les configurations atypiques ou qu'ils aient des besoins d'accessibilité spécifiques. Les options incluent :

- Personnalisation des contrôles : Attribuez chaque mouvement et action du personnage à la touche de votre choix.
- Revenir aux contrôles de base : Réinitialisez les contrôles aux paramètres par défaut du jeu en un seul clic.

Ces nouvelles fonctionnalités vous donnent le pouvoir de modeler l'expérience de jeu à votre image, vous permettant de vous immerger pleinement dans l'univers du jeu. Elles témoignent de notre engagement à créer un jeu flexible, accessible et gratifiant, répondant aux attentes et aux besoins de tous les joueurs.(Annexe 9)

En plus de ces deux menus, un bouton d'accès direct au menu de personnalisation des contrôles a été ajouté au menu de pause.

L'intégration de ces nouvelles fonctionnalités vise à rehausser considérablement la satisfaction des joueurs. Vous pourrez non seulement célébrer vos victoires avec style, mais aussi façonner votre expérience de jeu selon vos envies. Nous sommes convaincus que ces ajouts enrichiront considérablement l'univers du jeu et contribueront à sa pérennité auprès de notre communauté.

4.5 L'Intelligence Artificielle

À la dernière soutenance, nous avions intégré pour l'IA un système de Pathfinding lui permettant de suivre le joueur dans toute la maison. Pour ce faire, nous avons tout d'abord utilisé un NavigationMesh qui agit comme un calque sur le sol de la maison et qui permet d'indiquer à l'IA où elle peut se déplacer. Deuxièmement, nous avons utilisé un NavigationAgent3D qui sert de GPS à l'IA en lui indiquant le chemin à prendre pour atteindre le joueur le plus proche.

Nous avions également intégré le fait que lorsque l'IA attrape le joueur, elle se téléporte et téléporte le joueur à leur point de réapparition respectif, et elle fait perdre une vie au joueur. Nous avions donc ajouté le déclenchement d'un écran de Game Over dans le cas où un joueur n'a plus de vie.

Du côté du joueur, une nouvelle mécanique a été implémenté : celle de pouvoir se cacher pour échapper au fantôme. Cet ajout nous a donc requis de faire des modifications dans le script de l'IA, puisqu'il impacte le comportement du fantôme.

Pour ce faire, nous avons tout d'abord créé deux nouvelles propriétés : P1Hidden et P2Hidden, qui nous permettent d'accéder à l'attribut Hiden respectivement du joueur 1 et du joueur 2. Cet attribut, comme l'indique son nom, nous permet de savoir si le joueur est caché ou non.

Concernant le changement de comportement de l'IA, il va être impacté lorsqu'en multijoueur un des deux joueurs est caché et, quel que soit le mode de jeu, lorsque tous les joueurs sont cachés. Pour le premier cas, l'IA ne va uniquement cibler le joueur qui n'est pas caché actuellement. Et dans le deuxième cas, l'IA n'ayant plus aucun joueur à cibler, elle va simplement retourner à son point de réapparition en le prenant pour cible. Ainsi, la partie du script qui va être modifiée pour faire fonctionner cette nouvelle mécanique est celle qui indique à l'IA sa destination, soit la fonction ClosestPlayer.

Nous l'avons donc modifiée de manière à ce que l'IA se comporte selon les différents cas présentés au-dessus :

```
private Node3D ClosestPlayer()
{
    if (_player2 == null)
    {
        if (P1Hidden)
        {
            return GetNode<Node3D>("/root/Node3D/SpawnPoint/f");
        }
        return _player;
    }

    if (P1Hidden && P2Hidden)
    {
        return GetNode<Node3D>("/root/Node3D/SpawnPoint/f");
    }
    if (P2Hidden)
    {
        return _player;
    }

    if (P1Hidden)
    {
        return _player2;
    }

    if (this.GlobalPosition.DistanceTo(_player.GlobalPosition) <
        this.GlobalPosition.DistanceTo(_player2.GlobalPosition))
    {
        return _player;
    }
    else
    {
        return _player2;
    }
}
```

Cette première section gère le cas où il n'y a qu'un seul joueur dans la partie :

```
if (_player2 == null)
{
    if (P1Hidden)
    {
        return GetNode<Node3D>("/root/Node3D/SpawnPoint/f");
    }
    return _player;
}
```

Et le reste de la fonction gère les cas où nous sommes en multijoueur :

```
if (P1Hidden && P2Hidden)
{
    return GetNode<Node3D>("/root/Node3D/SpawnPoint/f");
}
if (P2Hidden)
{
    return _player;
}

if (P1Hidden)
{
    return _player2;
}
if (this.GlobalPosition.DistanceTo(_player.GlobalPosition) <
    this.GlobalPosition.DistanceTo(_player2.GlobalPosition))
{
    return _player;
}
else
{
    return _player2;
}
```

Comme annoncé à la dernière soutenance, nous avons implémenté un screamer pour l'IA. Pour réaliser cette étape, nous l'avons découpée en trois parties afin de n'oublier aucun élément et d'avoir une idée précise de comment nous allions nous y prendre :

1. La partie visuelle du screamer
2. La partie audio du screamer
3. Les transitions

Pour la première partie, nous avons fait le choix de faire un screamer visible en multijoueur par l'autre personne, c'est-à-dire que le screamer ne se contente pas d'une simple image, c'est l'IA que l'on voit en jeu qui va réaliser elle-même l'effet de surprise. (Annexe 10)

Pour ce faire, nous avons tout d'abord fait monter l'IA au niveau de la tête du joueur, afin que le visage du fantôme puisse être bien visible et au centre du champ de vision de sa caméra. Ensuite, nous avons immobilisé le joueur et l'IA afin de pouvoir exécuter le screamer. Cela a été réalisé en désactivant, dans un premier temps, la fonction `_PhysicsProcess` dans le script du joueur et de l'IA ainsi que la fonction `_Input` uniquement pour l'IA, ce qui a pour but de désactiver les contrôles pour le joueur et de désactiver la recherche pour l'IA. Pour le joueur, il a fallu additionnellement désactiver la fonction `_Process`, car elle est responsable du mouvement de la caméra. Cela est nécessaire dans notre cas, car comme évoqué précédemment, nous voulons faire en sorte que le screamer soit fait directement par l'IA, et contrôler la caméra est essentiel pour ce qui suit. Enfin, pour créer l'effet de surprise, nous pointons la caméra du joueur en direction du fantôme, permettant d'avoir la tête du fantôme en plein milieu du champ de vision du joueur et assurant l'effet de surprise. (Annexe 11)

Pour la seconde partie, nous avons décidé de faire en sorte que le screamer puisse être entendu par l'autre joueur, afin que ce dernier puisse identifier où son coéquipier s'est fait attraper et donc identifier où se situe le fantôme. Pour cela, il a fallu instancier un nœud `AudioStreamPlayer3D` dans la scène du fantôme afin de permettre à celui-ci d'émettre du son dans l'espace, plus ou moins audible par les joueurs en fonction de leur distance avec l'IA. Pour jouer le son strident du screamer, nous avons appelé la fonction `Play` sur ce nœud dans le script du fantôme lorsque celui-ci détecte qu'il a atteint un joueur. Nous avons jugé intéressant d'intégrer cette mécanique au jeu car elle renforce son côté coopératif et donne un vrai intérêt à jouer à plusieurs.

Concernant la troisième partie, nous avons choisis d'intégrer des transitions au niveau du screamer pour avoir un meilleur rendu visuel. De plus, en ayant la réapparition du joueur qui ne se fait pas brusquement, cela lui garantit une expérience plus plaisante en lui laissant le temps de reprendre ses esprits et lui permettant ainsi de mieux continuer la partie. Pour ce faire, nous avons premièrement intégré dans la scène de l'IA et du joueur un nœud « Timer » afin que le screamer puisse rester à l'écran un minimum de temps avant de téléporter l'IA et le joueur.

Pour se faire, nous déclenchons dans leurs scripts respectifs, leur « Timer » avec la fonction `Start` et nous relions le signal « `timeout` » intégré au « `Timer` » de Godot aux fonctions :

- `TimerEnded` dans le script de l'IA
- `EndScreamer` dans le script du joueur

Deuxièmement, nous avons intégré dans la scène du joueur un nœud « `AnimationPlayer` », nous permettant de faire un premier fondu de l'invisible vers le noir et un deuxième dans le sens inverse. Le premier est déclenché dès lors que le joueur est attrapé par l'IA et le deuxième est déclenché une fois que le « `Timer` » est écoulé, lorsque le screamer est terminé et que le joueur retourne à son point de réapparition. Nous avons également ajouté un second « `Timer` » nommé « `TimerPlayerDead` » pour gérer le cas particulier où un des deux joueurs meurt, lorsque son champ de vision tombe à zéro.

Dans ce cas présent, le compte à rebours dure 5.5 secondes, comparé aux 1.5 seconde habituelles, afin de laisser le temps au son du screamer de se terminer. De plus, l'écran de « Game Over » est affiché au lieu de déclencher le deuxième fondu mentionné auparavant. L'utilité de gérer ce cas est de garder une cohérence par rapport au fait que lorsque le joueur voit son champ de vision atteindre la valeur 0, il ne peut plus réapparaître et donc il ne peut y avoir de deuxième fondu.

En ce qui concerne le principe du déclenchement du screamer, il s'agit du même que celui utilisé pour téléporter l'IA et le joueur à leur point de réapparition respectif dès lors que l'IA a atteint le joueur, et que nous avions expliqué en détail dans le précédent rapport : c'est l'utilisation de signaux. Ici, nous n'en avons pas créé de nouveau, cela n'étant pas nécessaire. En effet, nous avons réutilisé le même que celui utilisé pour la mécanique de pathfinding, précédemment décrite, le screamer se déclenchant selon les mêmes conditions. Ainsi, tous les procédés qui viennent d'être détaillés ont pour origine la fonction `_TargetReached` située dans le script de l'IA relié au signal `target_reached` du nœud `NavigationAgent3D`, qui s'active une fois le pathfinding terminé.

Ainsi, au sein du code, toutes ces étapes se déclenchent dans l'ordre suivant :

- Le `NavigationAgent3D` détecte que la cible, ici le joueur, est atteinte. Cela va activer la fonction `_TargetReached` dans le script du fantôme, grâce au signal `target_reached`. Cette fonction va tout d'abord regarder si les 2 joueurs ne sont pas cachés :

```
private void _TargetReached()
{
    if (! (P1Hidden && _player2 == null) || (P1Hidden && P2Hidden))
        // Cas où les deux joueurs ne sont pas cachés
    {
        // code inutile dans l'explication
    }
    else
    {
        GD.Print("Players hidden");
    }
}
```

- Si les deux joueurs ne sont pas cachés, `_TargetReached` va alors tester quel joueur a été attrapé en regardant lequel est le plus proche de l'IA, ce qui a du sens, car notre IA suit le joueur le plus proche :

```
if (ClosestPlayer() == _player)
{
    // code inutile dans l explication
}
else if (ClosestPlayer() == _player2)
{
    // code inutile dans l explication
}
```

- Ensuite, elle appelle la fonction `GotCaught` en précisant en paramètre de cette dernière le joueur attrapé.

```
if (ClosestPlayer() == _player)
{
    // code inutile dans l explication
    GotCaught(_player);
}
else if (ClosestPlayer() == _player2)
{
    // code inutile dans l explication
    GotCaught(_player2);
}
```

- La fonction `GotCaught` va éléver le fantôme au niveau de la tête du joueur. Elle va envoyer un signal aux joueurs pour leur indiquer que l'un d'entre eux s'est fait attraper par le fantôme, en précisant la position du fantôme et la référence du joueur attrapé. Elle va jouer l'audio du screamer. Elle va désactiver la recherche du fantôme en désactivant la fonction `_PhysicsProcess`. Et elle va lancer en parallèle un compte à rebours qui déclenchera la fin du screamer pour le fantôme :

```
private void GotCaught(player p)
{
    Position = new Vector3(Position.X, Position.Y
        + 1.1f, Position.Z);
    EmitSignal(SignalName.Caught, GlobalPosition, p);
    _screamerSound.Play();
    SetPhysicsProcess(false);
    _timer.Start();
}
```

- Une fois le compte à rebours `_timer` de l'IA écoulé, celui-ci déclenche, grâce à son signal `timeout`, la fonction `TimerEnded` qui va téléporter l'IA à son point de réapparition et réactiver son protocole de recherche en réactivant la fonction `_PhysicsProcess` :

```
private void TimerEnded()
{
    Position = _spawnPoint;      //On TP le fantôme à son SpawnPoint
    Rpc("Synch_Pos", _spawnPoint);
    SetPhysicsProcess(true);
}
```

Nous nous situons désormais dans le script du joueur :

- Le signal `Caught` envoyé depuis la fonction `GotCaught` située dans le script de l'IA déclenche la fonction `_onEnemyCaught` qui va premièrement vérifier si le joueur concerné est bien celui dans lequel elle se trouve :

```
private void _onEnemyCaught(Vector3 fantomePos, player p)
{
    if (p == this)
    {
        // code inutile dans l explication
    }
}
```

- Si c'est le cas, on enregistre la rotation de la caméra du joueur, on appelle la fonction `ToggleScreamer` qui va démarrer le screamer et on démarre le compte à rebours `_timer` qui est responsable de terminer le screamer et qui est connecté à la fonction `EndScreamer` :

```
_originalCamRot = _camera.Rotation;
ToggleScreamer(fantomePos);
_timer.Start();
```

- La fonction `ToggleScreamer` va désactiver les contrôles liés aux interactions, aux mouvements et à la caméra du joueur en désactivant les fonctions `_Input`, `_PhysicsProcess` et `_Process`. Et elle va ensuite orienter la caméra en direction du fantôme pour avoir la partie visuelle du screamer et elle lance la première partie de la transition (le fade in) :

```
private void ToggleScreamer(Vector3 fantomePos)
{
    // code inutile dans l explication
    SetProcessInput(false);
    SetPhysicsProcess(false);
    SetProcess(false);
    _camera.LookAt(fantomePos);
    _toggleTransition.Play("transition_in");
}
```

- Une fois `_timer` écoulé, la fonction `EndScreamer` est déclenchée. Celle-ci va réduire de 1 le nombre de vies du joueur, représenté par son champ de vision, et elle va regarder si ce nombre a atteint 0. Pour gérer ce cas particulier, elle va déclencher le compte à rebours `_timerPlayerDead` qui va déclencher l'écran de Game Over :

```
private void EndScreamer()
{
    // code inutile dans l explication
    Champsdevision = 1;
    if (Champsdevision == 0)
    {
        _timerPlayerDead.Start();
    }
    else
    {
        // code inutile dans l explication
    }
}
```

- Si le joueur possède encore une ou plusieurs vies, on lance la seconde partie de la transition (le fade out), on téléporte le joueur à son point de réapparition, on assigne à la rotation de la caméra sa valeur originale et on réactive les contrôles pour les interactions, les mouvements et la caméra du joueur :

```
private void EndScreamer()
{
    // code inutile dans l explication
    if (Champsdevision == 0)
    {
        _timerPlayerDead.Start();
    }
    else
    {
        _toggleTransition.Play("transition_out");
        Position = _respawnPoint;
        _camera.Rotation = _originalCamRot;
        SetProcessInput(true);
        SetPhysicsProcess(true);
        SetProcess(true);
    }
}
```

Enfin, à la dernière soutenance, nous avions également prévu d'augmenter la vitesse de l'IA à chaque énigme résolue par le joueur. Cela a pour but d'accentuer la difficulté du jeu le long de la partie et donc de la rendre plus dynamique.

Cet ajout consiste en l'implémentation d'une fonction, nommée EnigmaFinished, qui va servir à augmenter la vitesse de l'IA.

```
private void EnigmaFinished()
{
    _speed += 1.0f / 3.0f;
}
```

Une fois la fonction créée, il faut désormais la relier aux énigmes. Pour ce faire, nous avons utilisé des signaux envoyés depuis les scripts des objets faisant partie des énigmes, car ce sont ces scripts qui indiquent si les énigmes sont terminées ou non. Nous avons donc créé dans chacun de ces scripts ce même signal :

```
[Signal]
public delegate void EnigmaFinishedEventHandler();
```

Ainsi, nous allons émettre ce signal dans la partie du script où l'on sait que l'énigme est terminée, soit la fonction CheckCollectedObjects. De plus, cette fonction étant exécuté à chaque image du jeu, il faut faire en sorte que le signal ne soit émis plus d'une fois. Pour éviter cela, nous avons créé un attribut booléen, `_enigmaFinished`, que nous mettrons à true lorsque l'énigme est terminée. Ainsi, avant d'émettre le signal, nous vérifions si l'énigme est déjà terminée ou non. On obtient donc la fonction suivante :

```
private void CheckCollectedObjects()
{
    if /*Condition pour que l'enigme soit terminée
        (varie en fonction de l'objet)*/
    {
        // code inutile dans l'explication
        if (!_enigmaFinished)
        {
            EmitSignal(SignalName.EnigmaFinished);
            _enigmaFinished = true;
        }
    }
}
```

Il ne reste désormais plus qu'à connecter ces signaux à la fonction EnigmaFinished, dans la fonction _Ready dans le script de l'IA :

```
public override void _Ready()
{
    // code inutile dans l explication
    GetNode<Area3D>("/root/Node3D/poupee").Connect("EnigmaFinished",
        new Callable(this, "EnigmaFinished"));
    GetNode<Area3D>("/root/Node3D/Bougie_eteinte").Connect("EnigmaFinished",
        new Callable(this, "EnigmaFinished"));
    GetNode<Area3D>("/root/Node3D/Froggie").Connect("EnigmaFinished",
        new Callable(this, "EnigmaFinished"));
    // code inutile dans l explication
}
```

4.6 Le Multijoueur & Réseau

L'expérience de jeu est souvent enrichie par la présence de plusieurs joueurs. C'est pourquoi la mise en place d'un mode multijoueur simple et accessible est un aspect crucial du développement de jeux vidéo. Dans ce contexte, nous allons explorer l'implémentation d'un mode multijoueur LAN (Local Area Network) dans Godot, un moteur de jeu réputé pour sa facilité d'utilisation et ses outils multijoueurs intégrés.

Pour la connexion entre les joueurs, nous avons opté pour le mode LAN, qui offre une solution simple et efficace pour les parties locales. Ce choix s'explique par plusieurs avantages :

- La simplicité de configuration : Le mode LAN ne requiert aucune infrastructure complexe ni de connaissance approfondie en réseautique. Les joueurs n'ont qu'à se connecter au même réseau local pour établir la communication.
- La performance : Le mode LAN offre généralement une latence réduite et des performances optimales, car les échanges de données se font directement entre les ordinateurs des joueurs.
- L'accessibilité : Le mode LAN est accessible à tous les joueurs disposant d'une connexion réseau locale, sans nécessité d'une connexion internet active.

Pour synchroniser les actions et les informations entre les joueurs, nous avons recours aux outils multijoueurs intégrés à Godot. Ces outils offrent un ensemble de fonctionnalités robustes pour gérer les communications réseau et la cohérence du jeu dans un environnement multijoueur.

La première étape consiste à vérifier que les joueurs sont connectés au même réseau local ou à un réseau local virtuel (VPN). Godot fournit des fonctions permettant de détecter les interfaces réseau disponibles et de s'assurer qu'elles appartiennent au même réseau.

Une fois la présence d'un réseau local confirmée, le joueur qui souhaite héberger la partie doit partager son adresse IPV4 avec les autres joueurs. Cette adresse permettra aux autres joueurs de se connecter à la partie en cours.

Pour synchroniser les actions et les informations entre les joueurs, nous allons utiliser la fonction RPC (Remote Procedure Call) de Godot. Cette fonction permet d'appeler une méthode ou une fonction spécifique sur tous les clients connectés à la partie. Cela permet de diffuser des informations importantes, telles que la position des personnages, les changements d'état du jeu ou les interactions entre les joueurs.

Le code correspondant est présenté ci-dessous.

```
private void _on_start_pressed()
{
    Rpc("StartGame");
}
[Rpc( MultiplayerApi.RpcMode.AnyPeer , CallLocal =
      true , TransferMode = MultiplayerPeer.TransferModeEnum.Reliable )]
private void StartGame()
{
    foreach (var player in Multi_game_manager.Players)
    {
        GD.Print(player.Name + " is playing"); //for debug
    }
    var scene = ResourceLoader.Load<PackedScene>
        ("res://Gameplay_les_differentes_scenes/scene.tscn").
        Instantiate<Node3D>();
    GetTree().Root.AddChild(scene);
    this.Hide();
}
```

La synchronisation du fantôme repose sur une communication entre l'hôte et les clients. L'hôte envoie régulièrement la position du fantôme aux clients, permettant à chaque joueur de visualiser sa position en temps réel. Cependant, dans certains cas spécifiques, tels que lorsque le fantôme atteint sa cible, le client peut également transmettre une nouvelle position du fantôme. Cette approche garantit une cohérence optimale de l'état du jeu pour tous les participants.

Pour la synchronisation des mouvements des joueurs entre eux, des outils intégrés à Godot ont été utilisés. Afin d'optimiser l'utilisation de la bande passante et d'éviter une surcharge du réseau, la fréquence des mises à jour de position a été réduite. Cette optimisation a malheureusement pour conséquence de rendre les mouvements des joueurs moins fluides, et apparaissant saccadés.

Le code suivant détaille la mise en œuvre de cette fonctionnalité :

```
public override void _PhysicsProcess( double delta )
{
    if ( GetNode<MultiplayerSynchronizer>("MultiplayerSynchronizer").
        GetMultiplayerAuthority() == Multiplayer .GetUniqueId() )
    {
        ( ... )
        MoveAndSlide();
        syncPos = GlobalPosition ;
        syncRot = GlobalRotation .Y;
        syncHead = GetNode<Node3D>("head ").GlobalRotation .X;
    }

    else
    {
        GlobalPosition = GlobalPosition .Lerp( syncPos , 0.25 f ) ;
        GlobalRotation = GlobalRotation with {
            Y = Mathf .Lerp( GlobalRotation .Y, syncRot , 0.25 f ) ;
        }
        GetNode<Node3D>("head ").GlobalRotation = GlobalRotation with {
            X = Mathf .Lerp( GetNode<Node3D>("head ").GlobalRotation .X, syncHead ,
                0.25 f ) ;
        }
    }
}
```

Afin de préserver la fluidité de l'expérience de jeu, il a été décidé de maintenir la pause pendant la synchronisation des menus. Cela signifie qu'à l'ouverture du menu pause, le jeu sera mis en pause pour les deux joueurs simultanément, et la même chose se produira à la fermeture du menu.

Pour le menu Game Over, la synchronisation est simplifiée car la mort d'un joueur entraîne la fin de la partie pour l'autre. Ainsi, il suffit de synchroniser l'ouverture du menu Game Over pour que les deux joueurs en prennent connaissance en même temps.

Le relancement de la partie est également synchronisé pour éviter tout décalage ou erreur de synchronisation qui pourrait survenir en cas d'instanciation différente des nœuds de jeu. Cela garantit que les deux joueurs reprennent la partie au même moment et dans le même état du jeu.

En résumé, la stratégie de synchronisation adoptée pour les menus et le relancement de la partie garantit une expérience de jeu cohérente pour les deux joueurs.

4.7 Le Son

Lors de la première soutenance, l'aspect sonore du jeu se limitait au menu d'accueil. Cette version enrichie intègre une bande son complète, composée de musiques et d'effets sonores, pour l'ensemble des menus et du jeu en lui-même.

Afin de créer une ambiance immersive et cohérente avec le thème horrifique du jeu, nous avons sélectionné des musiques libres de droit sur YouTube. Le choix s'est porté sur des morceaux à la fois intrigants et angoissants, aptes à susciter une sensation de peur et de tension chez le joueur.

Le moteur de jeu Godot ne prend pas en charge nativement le format MP3. Pour les intégrer au jeu, il a été nécessaire de convertir chaque fichier MP3 en format WAV. Cette conversion a été réalisée à l'aide du site web Convertio¹³.

Pour chaque scène du jeu, un nœud AudioStreamPlayer a été ajouté. Ce dernier a ensuite été lié au fichier audio correspondant à l'ambiance et au contexte de la scène. (Annexes 12 et 13)

L'intégration réussie de musique et d'effets sonores joue un rôle crucial dans l'immersion du joueur. Elle renforce l'impact de l'expérience horrifique et contribue à créer une atmosphère oppressante et dérangeante, comme évoqué précédemment dans la partie relative à l'intelligence artificielle. La bande son participe ainsi à l'accomplissement des objectifs du jeu en amplifiant les émotions ressenties par le joueur et en le plongeant au cœur de l'univers horrifique.

4.8 Le Site Web

Dans le cadre de la refonte de notre identité visuelle, nous avons entrepris une refonte complète de notre site web afin qu'il soit en parfaite adéquation avec notre nouveau logo (Annexe 14).

Cette démarche vise à créer une expérience utilisateur plus cohérente et harmonieuse, en reflétant fidèlement les valeurs et l'univers de notre marque.

La refonte du site web visait à :

- Assurer une cohérence visuelle : Le nouveau design du site web s'inspire du nouveau logo et adopte une palette de couleurs et des typographies en accord avec la charte graphique de la marque.
- Renforcer l'identité de marque : Le site web joue un rôle crucial dans la communication de l'image de marque. En adoptant un design cohérent et moderne, nous renforçons la perception de notre entreprise auprès du public cible.
- Améliorer l'expérience utilisateur : La navigation du site web a été optimisée pour une utilisation intuitive et fluide, en privilégiant une ergonomie claire et des fonctionnalités pratiques.

13. <https://convertio.co/fr/>

La refonte du site web d'Unununium constitue une étape importante dans la consolidation de notre identité visuelle et de l'image de marque. En offrant une expérience utilisateur plus cohérente, moderne et intuitive, nous renforçons notre présence digitale et contribuons à une meilleure perception de notre entreprise auprès du public.

En guise de première étape, nous avons procédé à une refonte complète de la page d'accueil du site. Dans le cadre de cette refonte, un nouveau logo a été créé pour notre jeu, comme illustré en Annexe 15.

Afin de captiver immédiatement l'attention de l'utilisateur, nous avons placé le titre du jeu en position centrale de l'écran, lui conférant une taille imposante. Le nom du jeu est accompagné de son slogan accrocheur, invitant l'utilisateur à découvrir l'expérience proposée. Un bouton clairement identifiable permet d'accéder directement à la page de téléchargement désormais fonctionnelle.

Voici le code HTML utilisé pour cette section :

```
<div class="home-hero">
    
    <h1 class="home-text4">Les maisons aux ames perdues</h1>
    <span class="home-text5">
        Decouvrirez-vous pourquoi les ames ne nous quittent pas
    </span>
    <a href="telechargement.html" class="home-navlink1 button">
        Telecharger maintenant
    </a>
</div>
```

Dans cette optique, nous avons conçu une page d'accueil sobre et minimaliste, tout en veillant à sa richesse informative (Annexe 16).

Notre page de téléchargement a fait peau neuve pour offrir une expérience utilisateur optimale. Désormais, elle arbore un design simple et intuitif, permettant aux utilisateurs de trouver et télécharger facilement le fichier correspondant au logiciel souhaité.

Des boutons clairement identifiés permettent de télécharger le fichier correspondant à chaque logiciel en un clic. Cette ergonomie simplifiée permet de gagner du temps et de faciliter l'accès aux logiciels recherchés.

La page de téléchargement est conçue pour offrir une navigation fluide et intuitive. La disposition des éléments et l'utilisation d'une typographie claire permettent aux utilisateurs de se repérer facilement et de trouver rapidement le logiciel qu'ils recherchent.

L'annexe 17 illustre la nouvelle page de téléchargement, mettant en lumière sa simplicité et son ergonomie intuitive. Les utilisateurs peuvent ainsi visualiser la disposition des éléments et la clarté des boutons de téléchargement.

Afin de créer une cohérence visuelle entre les différentes pages du site web, nous avons procédé à une harmonisation des couleurs en utilisant le langage CSS. Cette démarche vise à offrir aux utilisateurs une expérience visuelle fluide et agréable lors de la navigation sur le site.

Deux nouvelles pages ont été créées pour enrichir le contenu du site web :

1. Page Guide

Accessible depuis la barre de navigation principale, la page Guide permet aux joueurs de consulter directement le manuel d'installation du jeu depuis le site web, en cas de perte du manuel fourni avec le jeu physique. Cette initiative vise à faciliter l'accès à l'information et à offrir une meilleure expérience utilisateur.

2. Page Cadeau

La page Cadeau, accessible par un QR code présent uniquement dans la boîte du jeu accompagnée d'une lettre de remerciement, contient une vidéo humoristique en référence à une blague populaire dans le milieu informatique. Cette page, bien qu'ayant rencontré quelques difficultés techniques lors de sa création, apporte une touche de légèreté et de complicité avec les joueurs.

La vidéo "Rick Astley - Never Gonna Give You Up (Official Music Video)" a été intégrée à la page Cadeau. Cette vidéo, téléchargée au format mp4 depuis YouTube, est accompagnée d'une citation des droits d'auteur, conformément aux exigences de respect de la propriété intellectuelle.

Le cadeau est accessible via le lien suivant :

<https://unununium-company.github.io/UnununiumWeb/cadeau.html> (Annexe 18).

Les modifications apportées au site web d'Unununium visent à améliorer l'expérience utilisateur en offrant une navigation plus fluide, un accès direct à l'information utile et un contenu divertissant tout en respectant les droits d'auteur.

Le site web final est accessible via le lien suivant :

<https://unununium-company.github.io/UnununiumWeb/index.html> ou via notre QR code (Annexe 19).

5 Les problèmes rencontrés & les solutions apportées

5.1 Le Gameplay

Le choix du moteur de jeu Godot s'est avéré judicieux en ce qui concerne l'adaptation de notre jeu à divers systèmes d'exploitation. Grâce à ses fonctionnalités multiplateformes natives, nous avons pu compiler et déployer notre jeu sans encombre sur les plateformes ciblées, sans rencontrer de problèmes majeurs de compatibilité ou de performances.

L'intégration des animations dans notre jeu s'est également déroulée sans accroc, en grande partie grâce à la documentation complète et accessible de Godot. Les ressources pédagogiques et les exemples de code fournis nous ont permis de comprendre et d'implémenter efficacement les animations nécessaires, donnant vie à notre jeu et enrichissant l'expérience utilisateur.

En résumé, le processus d'adaptation du jeu aux différents systèmes d'exploitation et d'intégration des animations s'est déroulé de manière fluide et efficace grâce aux puissantes fonctionnalités et à la documentation détaillée de Godot.

Un problème récurrent concernait la réinitialisation des paramètres de contrôle par défaut à chaque ouverture du jeu. De plus, la personnalisation des contrôles était perdue lors de la fermeture et de la réouverture du menu des contrôles. Pour résoudre ce problème, nous nous sommes inspirés de la méthode utilisée par d'autres jeux, notamment les petits jeux, pour stocker les données utilisateur. Nous avons constaté que la plupart d'entre eux stockaient les paramètres dans un simple fichier texte. Nous avons donc décidé d'adopter la même approche. Le fichier de sauvegarde devait contenir les informations suivantes :

```
[keybinding]
```

```
forward="W"
back="S"
right="Q"
left="D"
space="Space"
accroupi="Shift"
sprint="A"
lumiere="E"
interact="F"
regene="R"
```

Afin d'accomplir cette tâche, nous avons développé un programme permettant l'instanciation d'une sauvegarde des touches. Les données sauvegardées sont stockées dans les données utilisateur du joueur. Au lancement du jeu, ce programme est exécuté et met à disposition plusieurs fonctionnalités. Détaillons la première d'entre elles :

```
public override void _Ready()
{
    config = new ConfigFile();
    if (!FileAccess.FileExists(SETTING_FILE_PATH))
    {
        config.SetValue("keybinding", "forward", "Z");
        config.SetValue("keybinding", "back", "S");
        config.SetValue("keybinding", "right", "Q");
        config.SetValue("keybinding", "left", "D");
        config.SetValue("keybinding", "space", "Space");
        config.SetValue("keybinding", "accroupi", "Shift");
        config.SetValue("keybinding", "sprint", "A");
        config.SetValue("keybinding", "lumiere", "E");
        config.SetValue("keybinding", "interact", "F");
        config.SetValue("keybinding", "regene", "R");
        config.Save(SETTING_FILE_PATH);
    }
    else
    {
        config.Load(SETTING_FILE_PATH);
    }
}
```

Lors du lancement du jeu, une vérification est effectuée afin de déterminer si un fichier de paramètres utilisateur existe déjà. Si tel est le cas, les données du fichier sont chargées à l'aide d'une fonction native de Godot. Dans le cas contraire, un nouveau fichier de paramètres est créé et initialisé avec les valeurs par défaut définies au début du projet. Ces valeurs par défaut prennent la forme d'un dictionnaire associant chaque action à une touche correspondante.

Une fonction dédiée permet d'établir un lien entre le menu du jeu et le fichier de paramètres. Cette fonction permet aux joueurs de sauvegarder les modifications qu'ils apportent aux paramètres de contrôle.

Code de la fonction de sauvegarde des paramètres :

```
public static void Save_keybinding(StringName action, InputEvent evente)
{
    string event_str = null;
    if (evente is InputEventKey eventKey)
    {
        event_str = OS.GetKeycodeString(eventKey.PhysicalKeycode);
    }
    else if (evente is InputEventMouseButton eventMouseButton)
    {
        event_str = "mouse_" +(eventMouseButton.ButtonIndex);
    }
    config.SetValue("keybinding", action, event_str);
    config.Save(SETTING_FILE_PATH);
}
```

La présente fonction assure la mise en correspondance entre un événement, ou plus précisément une action, et une touche spécifiée par l'utilisateur. Cette association est ensuite enregistrée de manière permanente dans le fichier approprié.

Afin que les utilisateurs puissent accéder à ces touches personnalisées dans le menu de contrôle, une fonction dédiée a été créée. Cette fonction effectue le chargement des touches définies et leur intégration au menu, les rendant ainsi opérationnelles. Le code source de cette fonction est présenté ci-dessous :

```
public static Dictionary<string, InputEvent> Load_keybinding()
{
    var keybinding = new Dictionary<string, InputEvent>();
    var keys = config.GetSectionKeys("keybinding");
    foreach (string key in keys)
    {
        var eventStr = config.GetValue("keybinding", key).ToString();
        if (eventStr.Contains("mouse_"))
        {
            InputEventMouseButton input_event;
            input_event = new InputEventMouseButton();
            input_event.ButtonIndex = (MouseButton)TOINT(eventStr.Split("_")[1]);
            keybinding[key] = input_event;
        }
        else
        {
            InputEventKey input_event;
            input_event = new InputEventKey();
            input_event.Keycode = OS.FindKeycodeFromString(eventStr);
            keybinding[key] = input_event;
        }
    }
    return keybinding;
}
```

La fonction Load_keybinding lit les raccourcis clavier et souris à partir d'une configuration et les stocke dans un dictionnaire. Voici une description détaillée de son fonctionnement :

Un dictionnaire nommé raccourci est créé pour stocker les raccourcis sous forme de paires clé/valeur, où la clé est une chaîne de caractères représentant le nom du raccourci et la valeur est un objet InputEvent représentant l'événement d'entrée associé.

La méthode config.GetSectionKeys("keybinding") est appelée pour obtenir une liste de toutes les clés de raccourcis définies dans la section "keybinding" de la configuration. La fonction itère sur chaque clé de la liste obtenue. Pour chaque clé :

La valeur associée à la clé dans la configuration est récupérée sous forme de chaîne de caractères (evenementStr). Si la chaîne evenementStr commence par "mouse_", cela indique qu'il s'agit d'un événement de souris. Un nouvel objet InputEventMouseButton est créé. L'indice du bouton de la souris est extrait de la chaîne en divisant evenementStr sur le caractère underscore (_) et en convertissant la partie droite en entier. Cet indice est assigné à la propriété ButtonIndex de l'objet InputEventMouseButton. L'objet InputEventMouseButton est ajouté au dictionnaire raccourcis avec la clé correspondante.

Si la chaîne evenementStr ne commence pas par "mouse _", il s'agit d'un événement de clavier. Un nouvel objet InputEventKey est créé. Le code de la touche est déterminé en appelant OS.FindKeyCodeFromString(evenementStr). Ce code est assigné à la propriété Keycode de l'objet InputEventKey. L'objet InputEventKey est ajouté au dictionnaire raccourci avec la clé correspondante.

Une fois tous les raccourcis traités, le dictionnaire raccourcis est retourné par la fonction.

5.2 Le Game design

La création de ce jeu a présenté plusieurs défis techniques nécessitant des solutions innovantes pour garantir une expérience de jeu fluide et engageante. Ce document décrit les défis rencontrés et les solutions mises en œuvre pour les surmonter.

Un défi initial concernait la gestion des objets récupérés par le joueur. La création d'une variable pour chaque objet s'est avérée peu pratique et inefficace, notamment pour un grand nombre d'éléments. Pour optimiser la gestion, les objets ont été regroupés dans une liste. Cela a permis une meilleure organisation et un suivi simplifié des objets collectés.

Un autre défi concernait le système d'allumage des bougies. La solution initiale consistait à utiliser des variables booléennes pour chaque bougie, indiquant si elle était allumée ou non. Cependant, cette approche manquait de flexibilité et de scalabilité. Pour pallier à cela, un dictionnaire a été utilisé pour mapper les bougies éteintes à leurs versions allumées. Cette approche a permis une gestion plus efficace de l'état des bougies et de l'interaction du joueur avec elles.

Un élément crucial du jeu était la détection des interactions du joueur avec les objets environnants, notamment pour déclencher des événements spécifiques. Pour ce faire, une fonction héritée a été utilisée. Cette fonction s'exécute chaque fois que la visibilité d'un objet change, permettant ainsi de détecter des événements tels que l'ouverture d'une porte lorsque la clé correspondante est obtenue.

La création des textes des notebooks s'est avérée être une tâche particulièrement chronophage. Pour optimiser ce processus, une méthode efficace a été mise en place pour identifier les zones d'interaction avec les objets. Cette approche a permis d'éviter de faire disparaître tous les objets en une seule fois, réduisant ainsi considérablement le temps nécessaire à la création des textes.

Un défi majeur concernait la détection de la collecte de tous les objets à collectionner. La solution initiale consistait à vérifier manuellement si chaque objet avait été collecté. Cependant, cette approche était peu pratique et sujette à des erreurs. Pour résoudre ce problème, une variable a été créée pour stocker et suivre le nombre d'objets collectés. Cette variable est mise à jour chaque fois qu'un objet est collecté. Une fois que le nombre requis d'objets a été atteint, la récompense correspondante devient visible pour le joueur.

Les défis techniques rencontrés lors de la création de ce jeu ont nécessité des solutions innovantes et une optimisation constante des processus. Les approches décrites dans ce document ont permis de surmonter ces défis et de garantir une expérience de jeu fluide, engageante et cohérente.

5.3 Les Graphismes

Le développement des graphismes de notre jeu s'est déroulé de manière fluide et efficace, en s'appuyant sur une stratégie combinant l'utilisation judicieuse de modèles 3D gratuits et la création d'animations personnalisées. Cette approche nous a permis de gagner un temps précieux tout en maintenant un contrôle créatif optimal sur l'aspect visuel du jeu.

L'utilisation de modèles 3D gratuits a constitué un atout majeur dans le processus de développement. En effet, cette stratégie nous a permis d'accéder à un large éventail de ressources de qualité, représentant des éléments clés de l'environnement du jeu, tels que des décors, des objets et certains personnages. Intégrer ces modèles existants nous a libéré d'un travail de création fastidieux et chronophage, nous permettant ainsi de nous concentrer sur les aspects les plus spécifiques et créatifs du jeu.

Parallèlement à l'utilisation de modèles 3D gratuits, nous avons également consacré du temps à la création d'animations personnalisées pour certains personnages clés du jeu. Cette démarche nous a permis de doter ces personnages de mouvements et de comportements uniques, en accord avec leur rôle et leur importance dans le récit. En insufflant une telle vie à nos personnages, nous avons renforcé l'immersion du joueur et contribué à l'atmosphère générale du jeu.

En combinant l'utilisation de modèles 3D gratuits et la création d'animations personnalisées, nous avons réussi à développer des graphismes de qualité pour notre jeu, tout en respectant les contraintes de temps et de budget. Cette approche nous a permis de gagner en efficacité et de maintenir un contrôle créatif optimal sur l'aspect visuel du jeu, contribuant ainsi à l'expérience globale des joueurs.

5.4 Les Menus

Le volet création d'interfaces du projet s'est déroulé sans accroc majeur. Cette réussite s'explique par l'adoption de principes de conception éprouvés lors de la dernière soutenance. Cette approche méthodique a permis d'anticiper et de résorber efficacement les défis potentiels, garantissant ainsi un processus efficient.

Forts de notre expérience lors de la précédente soutenance, nous avons mis en œuvre une stratégie de conception reposant sur des principes éprouvés. Cette approche visait à minimiser les risques et à maximiser l'efficacité du processus de création d'interfaces.

Grâce à cette stratégie proactive, aucune difficulté majeure n'a nécessité d'approfondissement particulier. Les interfaces ont été conçues et réalisées dans les délais et conformément aux exigences du projet.

Le succès du volet création d'interfaces démontre l'efficacité d'une approche basée sur l'expérience et la méthodologie. En capitalisant sur nos acquis et en adoptant des principes de conception éprouvés, nous avons pu naviguer dans ce projet complexe avec assurance et atteindre les objectifs fixés. Cette expérience nous conforte dans l'idée que l'anticipation et la planification rigoureuses sont des éléments clés pour mener à bien des projets d'envergure.

5.5 L'Intelligence Artificielle

Les problèmes rencontrés pour l'IA n'ont concerné que le screamer, ce qui a du sens, car c'est la partie qui nous a nécessité le plus travail.

Le premier problème rencontré est survenu lorsque nous faisions des essais en mode multijoueur. En effet, lorsque l'un des deux joueurs se faisait attraper, celui-ci était correctement téléporté et le screamer s'effectuait sans souci. Cependant, du côté de l'autre joueur, celui-ci se faisait également téléporter, sa caméra changeait de direction et la transition composée des deux fondus s'affichait sur son écran également. La cause ici est qu'il n'y avait pas de distinction entre les deux joueurs. Ce problème a pour origine que lorsque l'IA détecte qu'elle a attrapé un joueur, elle va envoyer le signal Caught à tous les joueurs sans distinction. Et puisque ce signal est relié au script de la scène joueur, dès lors que l'on instancie deux joueurs en multijoueur, ceux-ci vont avoir le même script et vont donc tous deux être reliés à ce même signal. Ainsi, au sein du script des joueurs, lorsque ceux-ci reçoivent le signal de l'IA, ils ne peuvent pas savoir à qui il est adressé, et donc le protocole du screamer et de la téléportation se déclenche pour les deux instances.

Pour régler ce problème, il va fallu d'une façon ou d'une autre ajouter un moyen de faire la distinction entre les deux joueurs, et de n'activer le protocole du screamer et de la téléportation que dans le script du joueur attrapé par le fantôme.

Pour faire la distinction entre les deux joueurs, nous avons utilisé le principe de recherche de notre IA, qui est que celle-ci va poursuivre le joueur le plus proche d'elle. En ayant ce fonctionnement, on peut donc savoir que lorsque l'IA a attrapé un joueur, ce dernier sera obligatoirement le plus proche. Ainsi, lorsque l'IA a détecté qu'elle a attrapé un joueur grâce à la fonction _TargetReached, elle va vérifier dans cette fonction quel joueur a été attrapé avec la fonction ClosestPlayer, ce qui nous donne le code suivant :

```
private void _TargetReached()
{
    if (! (P1Hidden && _player2 == null) || (P1Hidden && P2Hidden))
        // Cas où les deux joueurs ne sont pas cachés
    {
        if (ClosestPlayer() == _player)
        {
            // code inutile dans l'explication
        }
        else if (ClosestPlayer() == _player2)
        {
            // code inutile dans l'explication
        }
    }
    else
    {
        GD.Print("Players hidden");
    }
}
```

Concernant le second point, la manière actuelle qui permet de faire comprendre aux joueurs que l'un d'entre eux s'est fait attraper et donc qu'il doit exécuter du code, est l'utilisation d'un signal. Ainsi, nous l'avons modifié pour qu'il puisse passer en paramètre la référence du joueur attrapé et nous avons également modifié la fonction `_onEnemyCaught` dans le script du joueur pour qu'elle puisse récupérer cette référence :

Avant :

- Script IA : public delegate void CaughtEventHandler(Vector3 fantomePos) ;
- Script Joueur : private void _onEnemyCaught(Vector3 fantomePos)

Maintenant :

- Script IA : public delegate void CaughtEventHandler(Vector3 fantomePos, player p) ;
- Script Joueur : private void _onEnemyCaught(Vector3 fantomePos, player p)

Cela permet ensuite aux joueurs de savoir s'il s'agit bien d'eux lorsque le signal leur faire signe qu'un joueur s'est fait attraper par le fantôme.

Nous avons donc modifié la fonction `GotCaught`, qui est responsable de l'envoie du signal, pour qu'elle puisse prendre en paramètre la référence du joueur et qu'elle puisse la mettre en argument du signal :

```
private void GotCaught( player p )
{
    EmitSignal( SignalName.Caught , GlobalPosition , p );
    // code inutile dans l explication
}
```

Cela nous a donc également nécessité de modifier une seconde fois la fonction `_TargetReached` qui appelle la fonction `GotCaught` :

```
private void _TargetReached()
{
    if (! (P1Hidden && _player2 == null) || (P1Hidden && P2Hidden))
        // Cas où les deux joueurs ne sont pas cachés
    {
        if (ClosestPlayer() == _player)
        {
            // code inutile dans l'explication
            GotCaught(_player);
            // code inutile dans l'explication
        }
        else if (ClosestPlayer() == _player2)
        {
            // code inutile dans l'explication
            GotCaught(_player2);
            // code inutile dans l'explication
        }
    }
    else
    {
        GD.Print("Players hidden");
    }
}
```

Nous nous situons désormais dans le script des joueurs.

Une fois le signal envoyé, les deux joueurs vont le recevoir et appeler la fonction `_onEnemyCaught`. Cette dernière va regarder si la référence passée en paramètre correspond bien à la référence dans laquelle elle se situe. Si oui, on exécute alors le screamer et la téléportation :

```
private void _onEnemyCaught(Vector3 fantomePos, player p)
{
    if (p == this)
    {
        ToggleScreamer(fantomePos);
        _timer.Start();
    }
}
```

Nous avons également eu l'idée pour résoudre ce problème d'utiliser 2 signaux, un pour chaque joueur en vérifiant leur ID afin de savoir à qui le signal est destiné. Cependant, nous avons jugé après réflexion que cela serait trop volumineux en termes de code et c'est donc pourquoi nous avons préféré choisir la version ci-dessus.

Le second problème rencontré se déroulait à la réapparition du joueur après l'exécution du screamer. En effet, pour effectuer la partie visuelle du screamer, nous avons dit précédemment qu'il faut que le fantôme se mette au niveau de la tête du joueur et que la caméra du joueur soit orientée en direction du fantôme, cela permettant d'avoir le visage du fantôme clairement affiché.

Le problème est que la rotation de la caméra dépend du joueur. En effet, le joueur a la possibilité de changer la direction de la caméra avec sa souris. Cependant, ce changement de direction n'est pas sans conséquence pour le joueur, puisqu'il est également censé le faire changer lui-même de direction. Ainsi, nous avons besoin que la caméra soit orientée dans la même direction que le joueur, car sinon il risque de marcher en direction d'un point qui n'est pas celui pointé par la caméra.

Pour résoudre ce problème, nous avons donc enregistré la rotation de la caméra du joueur avant que le screamer se déclenche, la caméra étant orientée dans la bonne direction à ce moment-là. Ensuite, on oriente la caméra vers le fantôme pour effectuer le screamer et lorsque le screamer est terminé, on réoriente la caméra.

On a donc créé un attribut `_originalCamRot` dans le script du joueur :

```
private Vector3 _originalCamRot;
```

Lorsque le joueur se fait attraper, on enregistre la rotation de sa caméra :

```
private void _onEnemyCaught(Vector3 fantomePos, player p)
{
    if (p == this)
    {
        // code inutile dans l'explication
        _originalCamRot = _camera.Rotation;
        // code inutile dans l'explication
    }
}
```

Et on réinitialise l'orientation de la caméra une fois le screamer fini :

```
private void EndScreamer()
{
    // code inutile dans l'explication
    else
    {
        // code inutile dans l'explication
        _camera.Rotation = _originalCamRot;
        // code inutile dans l'explication
    }
}
```

5.6 Le Multijoueur & Réseau

La persistance des connexions réseau est un aspect crucial pour une expérience de jeu multijoueur fluide. Dans notre projet, nous avons rencontré des problèmes de déconnexion des joueurs, perturbant le déroulement du jeu et nécessitant parfois un redémarrage manuel.

Pour remédier à ces déconnexions, nous avons mis en place un système de détection et de gestion des interruptions de connexion. Ce système s'appuie sur les signaux déjà présents dans le moteur Godot, qui permettent de notifier les événements liés à l'état de la connexion réseau.

Lorsqu'une déconnexion est détectée, notre système entreprend les actions suivantes :

1. Réinitialisation des Outils Multijoueurs : Tous les outils et fonctionnalités liés au mode multijoueur sont réinitialisés pour garantir un état cohérent du jeu.
2. Retour au Menu Principal : Le joueur est redirigé vers le menu principal.

Bien que notre système actuel traite efficacement les déconnexions, il existe encore des cas où le retour au menu principal peut échouer, obligeant le joueur à redémarrer manuellement le jeu.

5.7 Le Son

L'aspect sonore joue un rôle crucial dans l'expérience globale d'un jeu vidéo, immergeant le joueur dans l'univers du jeu et renforçant l'impact des actions et des interactions. Dans le cadre du développement de notre jeu, nous avons adopté une approche pragmatique et efficace pour l'intégration sonore, en nous appuyant sur des ressources en ligne facilement accessibles et en privilégiant une mise en place rapide et sans accroc.

Reconnaissant l'importance de l'efficacité et du gain de temps, nous avons choisi de suivre un tutoriel YouTube détaillé pour la mise en place des mécaniques sonores de notre jeu. Cette approche nous a permis de bénéficier d'une guidance claire et étape par étape, facilitant ainsi l'apprentissage et l'intégration des éléments sonores dans notre projet.

L'intégration sonore de notre jeu s'est déroulée de manière efficace grâce à l'exploitation d'un tutoriel YouTube de qualité. Cette approche pragmatique nous a permis de tirer parti de la richesse des ressources en ligne tout en privilégiant la rapidité et la simplicité de mise en place. Le résultat final est une expérience sonore immersive qui contribue pleinement à l'univers du jeu.

5.8 Le Site Web

Comme évoqué précédemment, nous avons rencontré des difficultés lors de la création de la page "Cadeau" de notre site web. En effet, nous souhaitions intégrer une vidéo hébergée sur YouTube en utilisant le lien HTML fourni par la plateforme. Cependant, la vidéo ne se lançait pas malgré son affichage correct.

Malgré des recherches approfondies sur internet, nous n'avons pu identifier la cause du problème. Face à cette impasse, nous avons décidé de télécharger la vidéo et de la stocker sur le serveur de notre site web afin d'en garantir l'accessibilité.

Soucieux de respecter les droits d'auteur, nous avons cité la source de la vidéo dans le code HTML de la page. Nous avons également estimé que la vidéo étant facilement identifiable, une citation dans le code HTML était suffisante.

```
<video src="public/fichier/Rick Astley – Never Gonna Give You Up  
 ( Official Music Video ).mp4" controls autoplay width="800"  
 poster="public/external/icon.jpeg"  
 alt="https://www.youtube.com/watch?v=dQw4w  
 9WgXcQ&ab_channel=RickAstley">  
</video>
```

Pour donner suite à quelques défis techniques, nous avons finalisé la mise en place de notre page de cadeaux. Celle-ci est maintenant accessible à tous nos utilisateurs et nous vous invitons à en profiter pleinement.

6 Points en suspens et réflexions pour une version ultérieure

6.1 Le Gameplay

Dans cette partie, nous présentons une analyse détaillée du gameplay de notre jeu, en mettant en lumière les points forts et les axes d'amélioration potentiels. Nous tenons à souligner que l'ensemble de nos objectifs initiaux en matière de gameplay a été atteint, malgré les contraintes de temps et de documentation auxquelles nous avons été confrontés.

Points forts :

- Réalisation des objectifs de base : Le jeu parvient à offrir une expérience de base conforme aux objectifs initialement définis.
- Immersion dans l'univers du jeu : L'atmosphère et l'esthétique du jeu parviennent à transporter le joueur dans l'univers du jeu de manière convaincante.
- Variété des défis : Le jeu propose une diversité de défis et d'objectifs, permettant de maintenir l'intérêt du joueur sur la durée.

Axes d'amélioration :

1. Optimisation des escaliers

Le système d'escaliers actuel constitue un point d'insatisfaction majeure pour l'équipe. En raison des contraintes de temps et de documentation, la méthode implémentée n'est pas à la hauteur de nos ambitions. Nous sommes convaincus qu'avec des ressources supplémentaires, nous aurions pu proposer une solution plus aboutie et immersive.

2. Ajout d'un menu de contrôle détaillé

L'intégration d'un menu de contrôle plus complet aurait permis aux joueurs de personnaliser davantage leur expérience de jeu. Ils auraient pu ajuster des paramètres tels que la taille de l'écran, les effets sonores et la musique, selon leurs préférences.

3. Implémentation de nouvelles techniques de déplacement

Dans une optique d'enrichir le gameplay, nous envisageons d'introduire de nouvelles techniques de déplacement dans la prochaine version du jeu. Cela pourrait inclure la possibilité de grimper aux murs à l'aide d'échelles ou de glisser sur le sol en courant. Ces ajouts visent à diversifier les interactions des joueurs avec l'environnement et à offrir une expérience plus dynamique.

Malgré les contraintes rencontrées, nous sommes parvenus à réaliser un jeu répondant aux objectifs initiaux en matière de gameplay. Cependant, nous sommes conscients du potentiel d'amélioration de certains aspects. Les points soulevés dans ce document serviront de base à l'élaboration d'une future mise à jour majeure, qui visera à enrichir l'expérience de jeu et à répondre davantage aux attentes des joueurs.

6.2 Le Game design

Cette partie présente une analyse approfondie des axes d'amélioration potentiels pour un jeu vidéo. Il s'articule autour de trois points clés : la complexité des énigmes et des interactions, la diversification des environnements et des défis, et l'intégration de mécanismes de coopération.

La complexité des énigmes et des interactions :

- Des énigmes plus sophistiquées : Introduire des mécanismes de puzzle plus élaborés, nécessitant une réflexion approfondie et une analyse logique pour les résoudre.
- Des interactions enrichies : Permettre aux joueurs d'interagir de manière plus riche avec l'environnement, en utilisant des combinaisons d'objets pour résoudre des énigmes ou en découvrant des indices subtils pour progresser.
- De la narration immersive : Intégrer des dialogues et des éléments narratifs qui guident les joueurs dans leur exploration et enrichissent l'histoire du jeu.

La diversification des environnements et des défis :

- Des environnements variés : Proposer des lieux multiples et distincts à explorer, chacun avec ses propres énigmes, obstacles et défis uniques.
- De l'exploration immersive : Intégrer des éléments tels que des changements climatiques dynamiques ou des environnements souterrains complexes pour stimuler l'engagement des joueurs.
- Une progression stimulante : Maintenir l'intérêt en introduisant de nouveaux défis et mécanismes de jeu tout au long de l'aventure.

L'intégration de mécanismes de coopération :

- La collaboration entre joueurs : Concevoir des énigmes nécessitant une coordination et une communication entre les joueurs pour les résoudre.
- Des compétences complémentaires : Introduire des personnages aux capacités uniques, obligeant les joueurs à combiner leurs forces pour progresser.
- Des objectifs communs : Mettre en place des défis qui encouragent les joueurs à travailler ensemble pour atteindre des objectifs partagés.

6.3 Les Graphismes

Afin d'immerger davantage le joueur dans l'univers du jeu, il est proposé d'augmenter la qualité des objets 3D. Cela peut se concrétiser par :

- L'amélioration des textures : Des textures plus fines et plus détaillées donneront aux objets un aspect plus réaliste et plus immersif.
- La modélisation plus précise : Des modèles 3D plus complexes et plus précis permettront de mieux représenter les détails des objets, les rendant plus réalistes.
- L'ajout d'effets visuels : L'ajout d'effets visuels tels que des reflets, des ombres et des animations dynamiques rendront les objets plus vivants et plus réalistes.

L'ajout d'un système d'ombres dynamiques apportera un réalisme supplémentaire à l'environnement du jeu. Les ombres des objets réagiront à la lumière ambiante et aux sources de lumière présentes dans le jeu, créant ainsi un rendu plus immersif et crédible.

Afin de faciliter l'interaction du joueur avec son environnement, il est proposé de mettre en place un code couleur pour les objets. Ce code couleur permettra de différencier les objets avec lesquels le joueur peut interagir de ceux qui sont purement décoratifs.

- Les objets interactifs : Une couleur vive et distinctive pourrait être utilisée pour identifier les objets avec lesquels le joueur peut interagir, tels que des objets à ramasser, des leviers à activer ou des personnages non-joueurs avec lesquels dialoguer.
- Les objets utiles : Une autre couleur, peut-être moins vive, mais tout aussi claire, pourrait être utilisée pour identifier les objets qui procurent un avantage au joueur, tels que des points de vie ou des bonus.

Pour souligner davantage l'importance de certains objets, il est proposé d'ajouter un contour visuel autour d'eux. Ce contour pourrait être de couleur vive et suffisamment épais pour être facilement repérable par le joueur, même dans un environnement complexe.

Ces améliorations graphiques contribueront à créer une expérience de jeu plus immersive, réaliste et agréable pour le joueur. Elles permettront de renforcer l'engagement du joueur dans l'univers du jeu et de le rendre plus intuitif à utiliser.

6.4 Les Menus

L'équipe a finalisé la création de l'ensemble des menus prévus dans le cahier des charges initial. Ces menus répondent aux besoins fonctionnels définis et offrent une navigation intuitive aux utilisateurs.

Néanmoins, comme évoqué précédemment, l'équipe envisage d'intégrer un menu de contrôle plus approfondi dans une version ultérieure. Ce menu permettrait aux utilisateurs d'accéder à des options de configuration et de personnalisation plus avancées, répondant ainsi aux besoins spécifiques d'utilisateurs chevronnés.

L'ajout de ce menu de contrôle s'inscrit dans une démarche d'amélioration continue visant à enrichir les fonctionnalités de l'application et à répondre aux attentes d'une plus large audience.

6.5 L'Intelligence Artificielle

La première chose que nous n'avons pas pu implémenter exactement comme on le voulait est la fonction ClosestPlayer, située dans le script de l'IA. En effet, dans le cas où les deux joueurs ne sont pas cachés nous allons calculer la distance entre l'IA et chaque joueur, les comparer et enfin sélectionner le joueur dont cette distance est la plus petite :

```
private Node3D ClosestPlayer()
{
    // code inutile dans l explication
    if (this.GlobalPosition.DistanceTo(_player.GlobalPosition) <
        this.GlobalPosition.DistanceTo(_player2.GlobalPosition))
    {
        return _player;
    }
    else
    {
        return _player2;
    }
}
```

Le problème de cette implémentation est qu'elle calcule la distance entre les joueurs et l'IA sans prendre en compte les obstacles les séparant. Nous avions essayé d'implémenter cette fonction en calculant le vecteur le plus court pour atteindre le joueur, c'est-à-dire le vecteur calculé par le NavigationAgent3D pour se rapprocher de plus en plus du joueur avec la fonction GetNextPathPosition, mais cela nous créait un bug lorsque les deux joueurs étaient trop près l'un de l'autre que nous n'avons pas réussi régler. Cependant, la structure de la maison ne rend pas problématique cette implémentation.

Le second point que nous aurions aimé implémenter est que lorsque le screamer se déclenche, non seulement la caméra du joueur se tourne vers le fantôme, mais le corps du joueur également. Le souci est que lorsque nous avons fait cet ajout, les problèmes que nous avions eu avec la caméra après que le screamer soit terminé, se produisaient également. Et malgré les tentatives de résolution avec une variable stockée, comme pour résoudre le problème de la caméra, nous n'avons pu aboutir à un résultat satisfaisant et donc cet ajout ne s'est pas fait.

6.6 Le Multijoueur & Réseau

Le jeu présente actuellement un mode multijoueur LAN, permettant aux joueurs de se connecter et de jouer sur un réseau local. Afin d'étendre l'expérience de jeu et d'attirer un public plus large, il est proposé d'intégrer un mode multijoueur en ligne.

Cette évolution stratégique permettra de répondre à plusieurs objectifs clés, tel que :

- Accroître la base de joueurs : Le mode multijoueur en ligne permettra de franchir les barrières géographiques et de toucher un public plus vaste, au-delà des limitations du mode LAN. Cela se traduira par une augmentation significative du nombre de joueurs potentiels, favorisant ainsi la croissance de la communauté et l'engagement des joueurs.
- Renforcer la connectivité sociale : Le mode multijoueur en ligne permettra aux joueurs de se connecter et de jouer avec des amis, même ceux qui se trouvent à longue distance. Cela favorisera l'interaction sociale, la création de liens et l'esprit de communauté, renforçant ainsi l'expérience de jeu globale.
- Enrichir l'expérience de jeu : Le mode multijoueur en ligne pourra introduire de nouvelles fonctionnalités et de nouveaux modes de jeu, impossibles à implémenter dans le cadre du mode LAN. Cela permettra de diversifier l'expérience de jeu, de maintenir l'intérêt des joueurs et de stimuler la jouabilité.

L'intégration d'un mode multijoueur en ligne représente une opportunité stratégique majeure pour le jeu. En permettant aux joueurs de se connecter et de jouer ensemble, quel que soit leur emplacement, cette évolution permettra d'accroître la base de joueurs, de renforcer la connectivité sociale et d'enrichir l'expérience de jeu globale. En adoptant une approche de mise en œuvre réfléchie, en mettant l'accent sur le développement technique, la promotion et la communication, ainsi que la gestion de la communauté, il est possible de garantir le succès de cette initiative et de propulser le jeu vers de nouveaux sommets.

6.7 Le Son

Afin d'offrir une expérience immersive et captivante à nos joueurs, nous avons le plaisir d'annoncer l'intensification de nos efforts sur deux aspects fondamentaux de l'environnement sonore de notre jeu vidéo : le design sonore et la composition musicale.

Nous nous engageons à enrichir l'univers sonore du jeu en y insufflant une multitude de détails réalistes et captivants. Cela se traduira par :

- Une variété accrue d'effets sonores : Chaque élément du jeu, des environnements aux actions des personnages, en passant par les interactions avec les objets, sera doté d'effets sonores uniques et soigneusement conçus.
- Une bande son dynamique et réactive : L'ambiance sonore évoluera en fonction des actions du joueur et des événements qui se déroulent dans le jeu, créant ainsi une expérience immersive et réactive.
- Une utilisation magistrale de l'audio 3D : Les effets sonores spatialisés plongeront les joueurs au cœur de l'action, leur permettant de localiser précisément les sources sonores et d'apprécier la profondeur et la dimension de l'environnement virtuel.

Pour accompagner les joueurs dans leur périple, nous allons composé une bande-son originale et exclusive. Celle-ci sera composée de :

- Thèmes musicaux uniques pour chaque zone et personnage : La musique contribuera à définir l'identité de chaque lieu et de chaque protagoniste, renforçant ainsi l'attachement émotionnel des joueurs à l'univers du jeu.
- Compositions mélodiques et orchestrales : Des morceaux riches et captivants rythmeront les aventures des joueurs, les emportant dans une épopée sonore inoubliable.
- Une collaboration avec des compositeurs talentueux : Nous avons fait appel à des compositeurs de renom pour créer une bande-son qui saura toucher le cœur et l'esprit des joueurs.

En combinant un design sonore immersif et une bande-son originale de haute qualité, nous visons à créer une expérience audio exceptionnelle qui élèvera notre jeu vidéo à un niveau supérieur. Nous sommes convaincus que ces enrichissements sonores contribueront à plonger les joueurs dans un univers captivant et inoubliable, leur offrant une aventure vidéoludique à la fois divertissante et émotionnellement marquante.

6.8 Le Site Web

Dans l'ensemble, nous sommes satisfaits de la complétude de notre site web actuel. Il répond aux objectifs fixés en début de projet et offre une expérience utilisateur fluide et informative. Cependant, nous sommes convaincus qu'il y a encore place à l'amélioration afin de le rendre encore plus dynamique et interactif.

Actuellement, la rubrique "Problèmes et solutions" de notre site web est alimentée manuellement par l'équipe chargée de la maintenance du site. Cela peut s'avérer chronophage et limiter la réactivité face aux nouveaux problèmes rencontrés par les utilisateurs.

Pour pallier à ces inconvénients, nous proposons de mettre en place une base de données SQL qui permettra aux membres de l'équipe de soumettre directement leurs problèmes et solutions via une interface dédiée sur le site web.

Les avantages de cette solution :

- Réduction de la charge de travail de l'équipe de maintenance du site web : En permettant aux membres de l'équipe de soumettre directement leurs contributions, nous libérerons du temps précieux pour l'équipe de maintenance qui pourra se concentrer sur d'autres tâches stratégiques.
- Amélioration de la réactivité : La soumission directe des problèmes et solutions permettra une prise en charge plus rapide et efficace des requêtes des utilisateurs.
- Enrichissement du contenu de la rubrique "Problèmes et solutions" : La participation active de l'ensemble de l'équipe permettra d'étoffer la base de connaissances et d'offrir aux utilisateurs une palette plus large de solutions à leurs problèmes potentiels.

Actuellement, les utilisateurs n'ont pas la possibilité de laisser des commentaires directement sur le site web. Cette fonctionnalité permettrait de :

- Recueillir les impressions et suggestions des utilisateurs : Les commentaires des utilisateurs constituent une source précieuse d'informations pour identifier les points forts et les points faibles du site web et ainsi orienter les futures améliorations.
- Renforcer l'interaction avec les utilisateurs : La possibilité de laisser des commentaires permettra de créer un lien plus direct avec les utilisateurs et de les impliquer davantage dans le développement du site web.
- Améliorer la satisfaction des utilisateurs : En prenant en compte leurs commentaires et en leur offrant la possibilité de s'exprimer, nous contribuerons à renforcer leur satisfaction et leur engagement envers notre site web.

En intégrant une base de données SQL pour la gestion des problèmes et solutions et en mettant en place un système de commentaires pour les utilisateurs, nous sommes convaincus que nous pourrons améliorer considérablement l'expérience utilisateur sur notre site web. Ces changements permettront de le rendre plus dynamique, interactif et mieux adapté aux besoins de notre communauté.

Les points à développer ultérieurement :

- Définir les modalités de gestion des commentaires des utilisateurs : Il sera nécessaire de déterminer les processus de modération et de validation des commentaires afin de garantir un contenu de qualité et respectueux.
- Explorer les différentes solutions techniques pour l'intégration des commentaires : Il existe différentes options techniques pour mettre en place un système de commentaires sur un site web. Il sera important de choisir la solution la plus adaptée à nos besoins et à notre infrastructure.
- Promouvoir la participation des utilisateurs : Il sera nécessaire de sensibiliser les utilisateurs à l'existence du système de commentaires et de les encourager à s'exprimer afin de recueillir un maximum de retours pertinents.

En mettant en œuvre ces améliorations, nous réaffirmons notre engagement à offrir à nos utilisateurs un site web de qualité, performant et répondant à leurs attentes.

7 Réflexion personnelle sur l'expérience du projet

7.1 Lucas Bernardeau

Dans cette partie du rapport, je souhaite partager mes réflexions personnelles sur le projet de jeu vidéo, en abordant à la fois les aspects positifs et négatifs de mon expérience. Mon objectif est de tirer des leçons de cette aventure et d'identifier les points à améliorer pour de futurs projets.

Rétrospectivement, je pense qu'il aurait été préférable de former les équipes en novembre plutôt qu'en septembre. En effet, à ce stade précoce de l'année scolaire, nous ne connaissons pas encore suffisamment nos camarades de classe. Un temps de découverte mutuelle aurait permis de constituer des groupes plus soudés et partageant des centres d'intérêt communs, favorisant ainsi une meilleure cohésion et une dynamique de travail plus productive.

L'utilisation de Godot comme moteur de jeu s'est avérée être un défi inattendu. La documentation et les ressources disponibles en C#, langage de programmation associé à Unity, étaient bien plus abondantes et accessibles. Cela a entraîné des difficultés considérables dans la création de chaque élément du jeu, nous obligeant à tout développer manuellement. L'utilisation de Unity, avec ses plugins déjà existants, aurait permis de gagner un temps précieux et de se concentrer davantage sur les aspects créatifs du projet.

Malgré les obstacles rencontrés, ce projet a été une source d'enrichissement personnel sur plusieurs plans :

J'ai eu l'opportunité de rencontrer et de collaborer avec des camarades dans un contexte différent de celui de la classe traditionnelle. Cette expérience a permis de tisser des liens et de découvrir de nouvelles personnalités.

Malgré les défis, j'ai éprouvé une grande satisfaction à travailler sur ce projet. L'esprit d'équipe, la créativité et la recherche de solutions ont été des moteurs de motivation tout au long de cette aventure. Je suis déterminé à poursuivre le développement de ce jeu avec certains membres de l'équipe, en tirant profit des leçons apprises et en explorant de nouvelles pistes pour le mener à bien.

Ce projet de jeu vidéo a été une expérience riche en apprentissages et en émotions. Si des points d'amélioration peuvent être identifiés, notamment en ce qui concerne la formation des équipes et le choix du moteur de jeu, les aspects positifs tels que la découverte de nouvelles personnes, l'immersion dans l'univers du jeu vidéo et la satisfaction personnelle l'emportent largement. Je suis convaincu que les leçons tirées de cette expérience nous permettront d'aborder de futurs projets avec encore plus de succès et de détermination.

7.2 Valentin BRAULT

En ce qui concerne mon ressenti sur le projet, j'ai trouvé la création de ce jeu autant intéressante d'un point de vue technique que d'un point de vue global sur le travail en groupe.

A propos de mon premier point, j'ai particulièrement aimé le fait de développer un jeu vidéo. Tout d'abord, étant passionné de jeux vidéo, pouvoir créer le sien en ayant aucune limite fut un réel plaisir pour moi. De plus, coder des fonctions qui ont une utilité que je peux directement constater donne du sens au développement, ce qui a créé chez moi de la motivation supplémentaire et a rendu ce processus plus palpitant. En outre, coder moi-même un jeu vidéo m'a permis de voir l'envers du décor et donc de mieux comprendre comment est-ce que les différents éléments d'un tel projet fonctionnent réellement. Cela m'a également permis de me mettre à la place des développeurs de jeux vidéo, pour qui ce que nous avons fait est leur quotidien, et donc d'être plus compatissant lorsque je rencontre un bug lors de mes sessions de jeu, comprenant désormais la difficulté du métier.

Concernant les points négatifs pour la partie technique, j'ai tout d'abord trouvé l'initiation à Godot un peu complexe et les premières heures de découverte et d'expérimentation n'étaient clairement pas les plus agréables. Cependant, pour remédier à cela, j'ai suivi le tutoriel de jeu en 3D sur le site officiel de Godot qui m'a, je l'admet, bien aidé à me familiariser avec le moteur de jeu. Une autre critique que je peux émettre sur cette partie est que Godot possédant son propre langage de programmation, conçu pour les débutants en développement, il est assez difficile de trouver des informations pour les utilisateurs de C#. Le site officiel de Godot possède certes une documentation plutôt complète sur ce langage, mais les tutoriels ne concernent majoritairement pas C# et la compréhension de GDScript m'a demandé un certain temps supplémentaire.

Pour ce qui est du second point, ce projet m'a énormément appris sur le travail en groupe. En effet, la création d'un jeu étant un processus ardu, il nous a demandé une rigueur dans l'organisation des tâches que je n'avais jamais connu auparavant. Cela m'a appris à me fixer des deadlines tout au long de l'année ce qui m'a rendu plus autonome et plus résistant face à la pression. De plus, cela m'a permis de me familiariser avec le concept de réunions, que nous avons organisées régulièrement. Enfin, ce projet m'a également instruit à m'adapter aux différentes personnalités.

En conclusion, je pense que ce projet est une très bonne initiation aux travaux de groupes que nous continuerons à rencontrer tant bien à EPITA que dans le monde du travail.

7.3 Marius VANHAESEBROUCKE

Ce projet a été l'occasion de rencontrer de nouveaux camarades et de tisser des liens en dehors du cadre scolaire classique. L'esprit d'équipe et la complémentarité des compétences ont permis de surmonter les obstacles et de créer une dynamique de travail productive.

Malgré les défis techniques, le projet a été une source d'apprentissage et d'épanouissement personnel. La satisfaction de participer à la création d'un jeu vidéo et de voir son travail prendre vie a été un moteur de motivation important.

L'expérience a renforcé l'envie de continuer à développer le jeu en tirant profit des leçons apprises et en explorant de nouvelles solutions.

La formation des équipes en septembre n'a pas permis de se connaître suffisamment avant de se lancer dans le projet. Cela a pu limiter la cohésion et la synergie entre les membres.

L'utilisation de Godot a posé des difficultés en raison du manque de ressources disponibles par rapport à Unity. Cela a entraîné un surcroît de travail et a ralenti le développement.

Ce projet a été une expérience enrichissante malgré les défis rencontrés. Les aspects positifs tels que la collaboration, l'apprentissage et la satisfaction personnelle prennent le pas sur les points négatifs. Les leçons tirées permettront d'aborder de futurs projets avec plus de succès et de détermination.

7.4 Célia BOCAGE

L'utilisation d'un moteur de jeu moins connu comme Godot a nécessité l'apprentissage de nouvelles techniques et a permis de développer des compétences en programmation et en résolution de problèmes.

Face aux difficultés rencontrées, l'équipe a dû faire preuve d'adaptation et de créativité pour trouver des solutions innovantes et contourner les obstacles.

La cohésion du groupe et la persévérance des membres ont permis de surmonter les défis et de mener le projet à terme.

La formation tardive des équipes et le choix du moteur de jeu ont pu ralentir le développement et complexifier certaines tâches.

L'utilisation de Godot a entraîné des difficultés inattendues et un temps de développement plus long que prévu.

Le prototype final ne reflète pas l'intégralité du potentiel du jeu et laisse entrevoir des améliorations possibles.

Ce projet a été un tremplin pour développer des compétences techniques, apprendre à s'adapter aux imprévus et renforcer l'esprit d'équipe. Si des améliorations peuvent être apportées à la planification et au choix des outils, les acquis de cette expérience permettront d'aborder de futurs projets avec plus de confiance et de réussite.

7.5 Guillaume JUILA

Ce projet a permis de se découvrir soi-même et de découvrir les talents et les personnalités des autres membres de l'équipe.

La création d'un jeu vidéo a offert un espace d'expression créative et artistique unique, permettant de concrétiser des idées et de partager une vision commune.

La réalisation d'un prototype fonctionnel a été une source de grande satisfaction personnelle et collective, récompensant les efforts et l'investissement de l'équipe.

Les difficultés liées au moteur de jeu et au manque de ressources ont pu générer des frustrations et ralentir la progression du projet.

Le temps imparti et les moyens disponibles n'ont pas permis de finaliser le jeu à son plein potentiel.

La communication au sein de l'équipe n'a pas toujours été optimale, ce qui a pu entraîner des incompréhensions et des retards.

Ce projet a été une aventure humaine et créative enrichissante, permettant de développer des compétences relationnelles, de s'exprimer à travers le jeu vidéo et de vivre la satisfaction de l'accomplissement collectif. Malgré les frustrations techniques et les limites du prototype, l'expérience a été positive et a permis de tisser des liens et de s'épanouir

8 Conclusion

Après six mois de développement intense et d'une implication sans faille, l'équipe Unununium est fière de présenter une première version aboutie de son jeu vidéo "Les maisons aux âmes perdues". Cette première étape marque le début d'une aventure prometteuse qui s'annonce riche en rebondissements et en découvertes.

Forts de leur expérience et de leur succès initial, les membres de l'équipe Unununium abordent l'avenir avec confiance et enthousiasme. Ils sont animés par une passion commune pour le jeu vidéo et par une détermination sans faille à offrir aux joueurs une expérience unique et inoubliable.

Le développement de "Les maisons aux âmes perdues" se poursuivra dans les mois à venir avec une rigueur et une passion constantes. L'équipe Unununium est déterminée à enrichir et à raffiner l'expérience de jeu pour offrir aux joueurs un univers captivant et immersif.

Les prochains mois seront marqués par un travail intense et une implication sans faille de l'ensemble de l'équipe. De nouvelles fonctionnalités seront implémentées, des éléments de gameplay seront peaufinés et l'univers du jeu sera enrichi. Chaque étape de développement sera l'occasion de nouvelles avancées et de nouvelles surprises pour les joueurs.

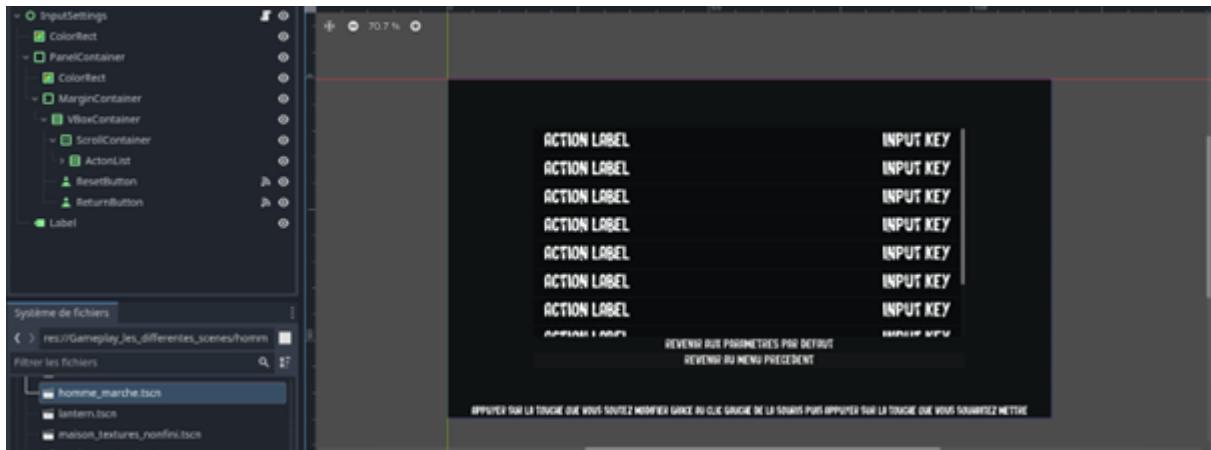
"Les maisons aux âmes perdues" est un jeu vidéo en constante évolution qui ne cesse de se perfectionner. L'équipe Unununium est à l'écoute des joueurs et s'engage à prendre en compte leurs remarques et suggestions pour offrir une expérience de jeu toujours plus optimale.

Grâce à l'engagement et au talent de l'équipe Unununium, "Les maisons aux âmes perdues" a toutes les chances de devenir un jeu vidéo incontournable. Les joueurs peuvent s'attendre à vivre une aventure captivante et inoubliable dans un univers fascinant et mystérieux.

Unununium : Une équipe passionnée qui crée des jeux vidéo d'exception.

9 Annexes

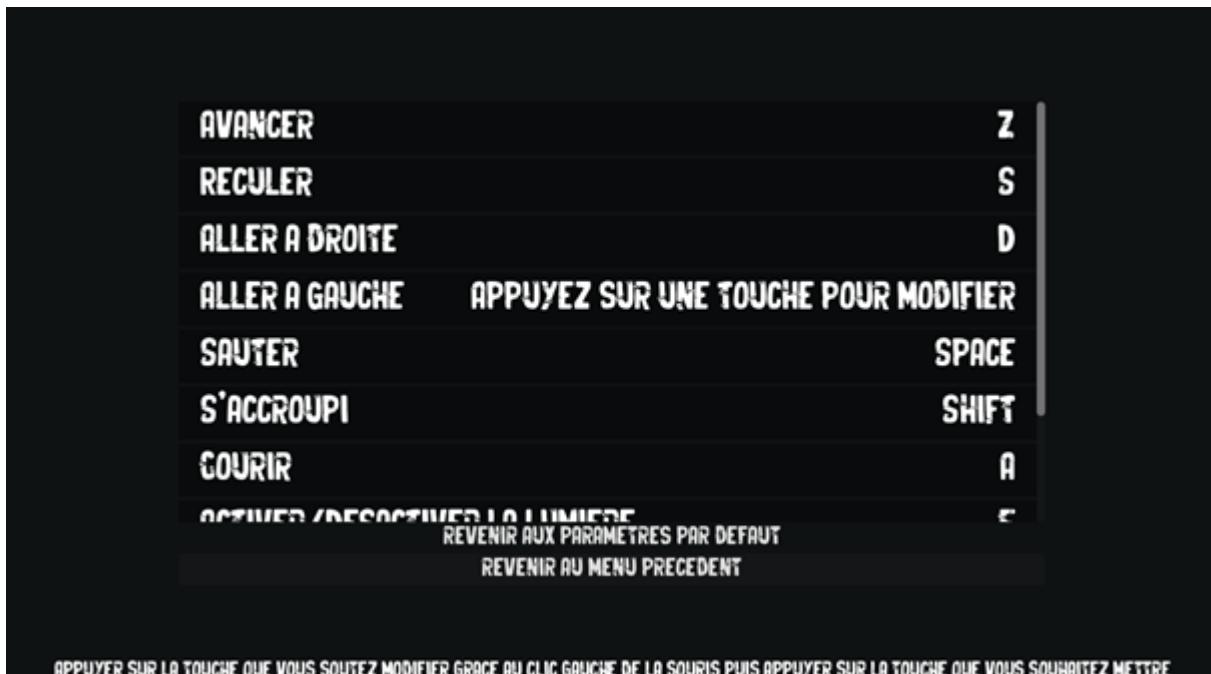
9.1 Annexe 1 : image illustrant la création de l'interface dans Godot



9.2 Annexe 2 : image illustrant l’interface utilisateur du menu de contrôle



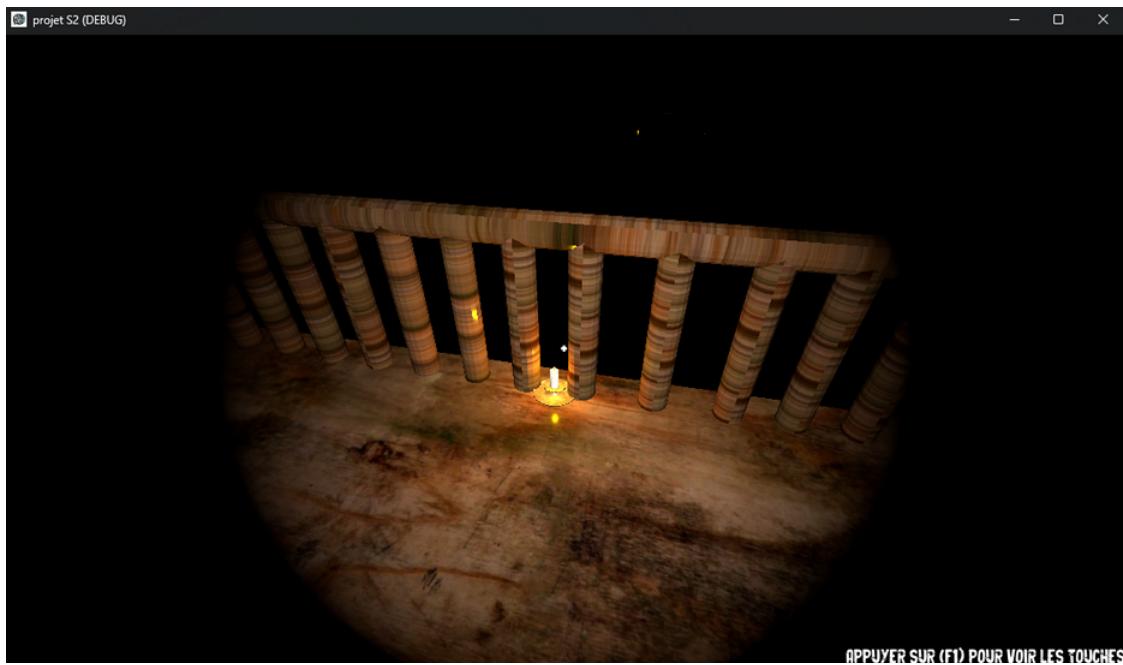
9.3 Annexe 3 : Illustration de l'état "Attente de modification de touche"



9.4 Annexe 4 : Bougie éteinte



9.5 Annexe 5 : Bougie allumée



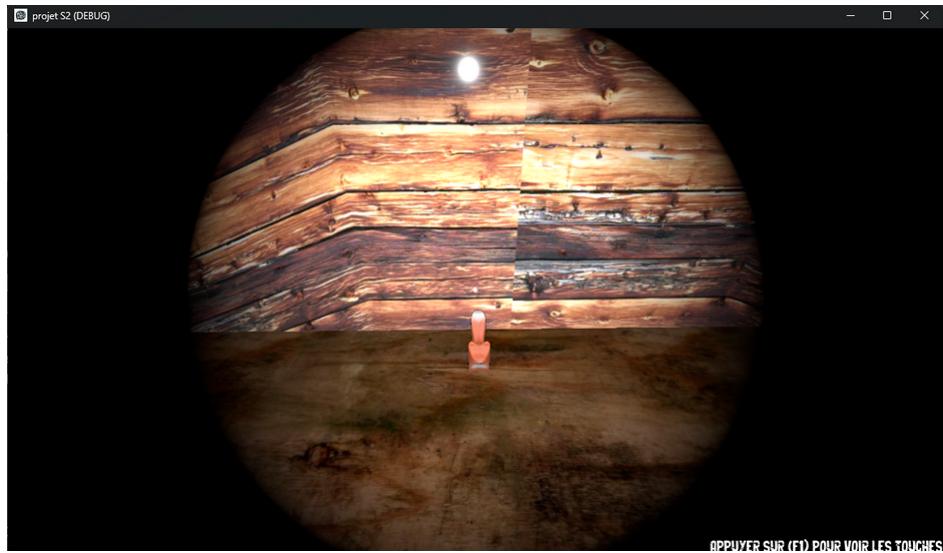
APPUYER SUR (F1) POUR VOIR LES TOUCHES

9.6 Annexe 6 : Une des poupees



APPUYER SUR (F1) POUR VOIR LES TOUCHES

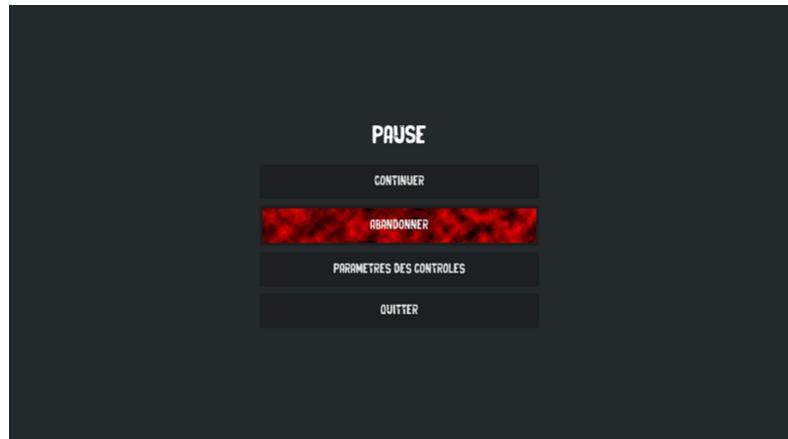
9.7 Annexe 7 : Un des jouets



9.8 Annexe 8 : Menu victoire



9.9 Annexe 9 : Menu pause



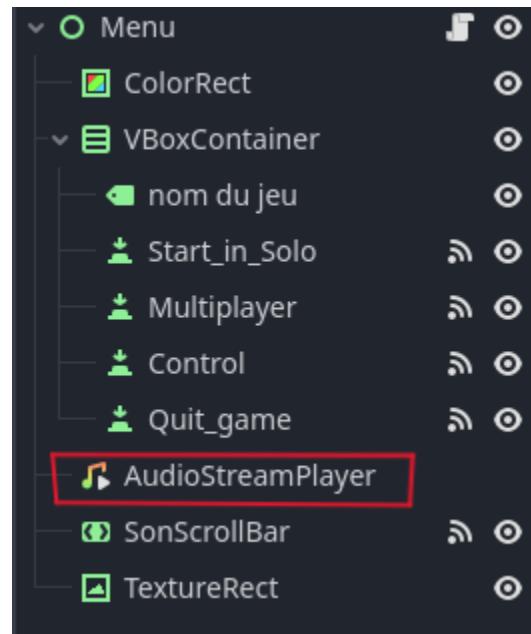
9.10 Annexe 10 : Screamer du point de vue extérieur



9.11 Annexe 11 : Screamer du point de vue du joueur attrapé



9.12 Annexe 12 : Nœud AudioStremPlayer dans l’arborescence d’une scène



9.13 Annexe 13 : Paramètre du nœud AudioStremPlayer



9.14 Annexe 14 : Nouveau logo de l'entreprise



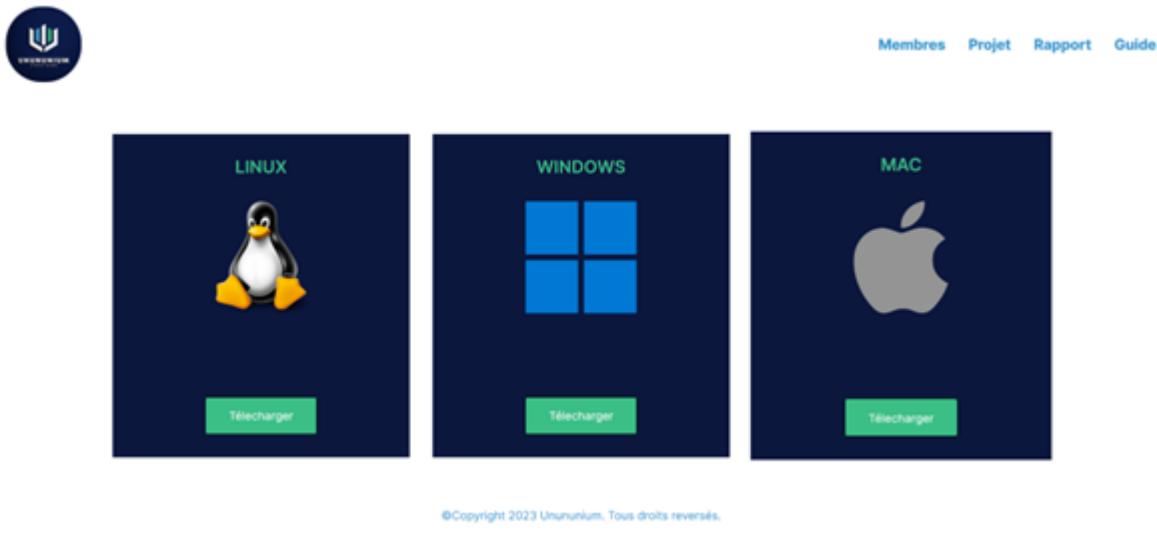
9.15 Annexe 15 : Icon de notre jeu vidéo



9.16 Annexe 16 : Image de notre page d'accueil

A screenshot of a website page. At the top left is a small circular logo. At the top right are navigation links: "Membres", "Projet", "Rapport", and "Guide". The main feature is a large, circular logo for the game "LES MAISONS AUX AMES PERDUES", which is identical to the one shown in Annex 15. Below the logo, the game's name is displayed in a large, blue, serif font. A subtitle below it reads "Découvrez-vous pourquoi les âmes ne nous quittent pas." A green button labeled "Vidéos et news" is visible. At the bottom of the page, there is a copyright notice: "©Copyright 2023 Unununium. Tous droits réservés."

9.17 Annexe 17 : Image de notre page de téléchargement



9.18 Annexe 18 : QR code vers la page-cadeau



9.19 Annexe 19 : QR code vers notre site internet

