



**ALTERAVITA**

# Rapport de Soutenance Intermédiaire

Réalisé par :

Hugo MEENS

Elya MAURY

Ethan PERRUZZA

Timothy DIDIERJEAN

Projet S2 - EPITA

# Table des matières

<b>1 Rappel du Projet</b>	<b>3</b>
1.1 Le Groupe . . . . .	3
1.1.1 Présentation . . . . .	3
1.1.2 Gestion Organisationnelle . . . . .	3
1.2 Histoire . . . . .	3
1.2.1 Synopsis . . . . .	3
1.2.2 Conclusion de l'histoire . . . . .	4
1.2.3 Intentions/Visée de réflexion du scénario . . . . .	4
1.3 Mécaniques de GamePlay . . . . .	5
1.3.1 Présentation Générale . . . . .	5
1.3.2 Les Énigmes . . . . .	6
1.3.3 Les Monstres . . . . .	6
1.3.4 Les Fioles . . . . .	7
1.3.5 Labyrinthe fléché . . . . .	7
<b>2 Rappel Des Objectifs pour la Soutenance 2</b>	<b>8</b>
<b>3 Avancement</b>	<b>9</b>
3.1 Graphismes . . . . .	9
3.2 Les Personnages . . . . .	13
3.3 Création des Salles Communes . . . . .	15
3.3.1 Les Cellules . . . . .	15
3.3.2 La Salle Principale . . . . .	15
3.4 Les monstres . . . . .	16
3.5 Potions . . . . .	23
3.6 Menu Principal . . . . .	27
3.7 Multijoueur en ligne . . . . .	29
3.7.1 Implémentation dans Unity . . . . .	29
3.8 Énigme : Le Labyrinthe de Boîtes . . . . .	29
3.8.1 Création des Objets pour l'Enigme . . . . .	29
3.8.2 Création de la Salle . . . . .	32
3.9 Énigme : Le Labyrinthe Invisible . . . . .	36
3.9.1 Modifications du labyrinthe . . . . .	36
3.9.2 Intégration du multijoueur en ligne . . . . .	36
3.9.3 Améliorations à venir . . . . .	37
3.10 Énigme : Labyrinthe fléché . . . . .	38
3.11 Narration . . . . .	41
3.11.1 Rédaction du scénario . . . . .	41
3.11.2 Système de Dialogue . . . . .	42
3.12 Site Web (Front) . . . . .	44
3.13 Site Web (Back) . . . . .	46
3.14 Launcher . . . . .	48
3.15 Résumé de l'Avancement . . . . .	49
3.16 Conclusion . . . . .	49

<b>4</b>	<b>Prévision</b>	<b>51</b>
<b>5</b>	<b>Ressources</b>	<b>52</b>
<b>6</b>	<b>Conclusion</b>	<b>53</b>

# Chapitre 1

## Rappel du Projet

Cette section a pour but de rappeler les éléments du cahier des charges dont nous allons parler dans le rapport.

### 1.1 Le Groupe

#### 1.1.1 Présentation

L'équipe en charge de la réalisation de ce projet est nommée XENOR et est constituée de :

**hugo.meens** : Hugo MEENS (chef de projet)

**elya.maury** : Elya MAURY

**ethan.perruzza** : Ethan PERRUZZA

**timothy.didierjean** : Timothy DIDIERJEAN

#### 1.1.2 Gestion Organisationnelle

Le groupe se réunit lors de réunions hebdomadaires tous les lundi soir ou mardi midi (suivant l'emploi du temps) pour faire un point sur l'avancement et les objectifs de la semaine.

Nous utiliserons les fonctionnalités « projects » et les « issues » de GitHub pour connaître l'avancement du projet : les tâches en cours, réalisées ou à réaliser. De plus, chacun indiquera sur un serveur Discord propre à l'équipe sur quoi il travaille en ce moment.

Le serveur Discord de l'équipe permettra également de discuter des éventuels problèmes rencontrés par chacun et de garder un lien hors des réunions hebdomadaires.

### 1.2 Histoire

#### 1.2.1 Synopsis

Dans un monde, où la médecine ne cesse d'évoluer, vous vous réveillez enfermé dans une cellule d'un laboratoire. Par chance, tout le monde n'a pas été complètement annihilé par le désir croissant des populations occidentales de contrôler leur santé et de repousser les limites de leur finitude. Et, alors que les scientifiques et les managers du laboratoire sont convoqués par le directeur du laboratoire Xenor pour discuter des prochains essais humains de transgénèse, c'est le moment que Clothilde, opératrice réseau choisit pour vous aider à vous échapper. Il faut dire que Clothilde n'est pas née de la dernière pluie, elle a été formée à l'EPITA, la plus grande école d'ingénieur en informatique de l'Univers.

Issue de la promo 2026, elle est la dernière épitéenne encore en vie à ce jour... En effet en 2052, les dirigeants des plus hautes sphères de la planète se firent pour mission d'éliminer tous ceux qui pourraient nuire à leur projet et la trop grande place de l'éthique chez les épitéens avait conduit à une

forte diminution du nombre d'Alumni. Clothilde était une autre personne à l'époque, elle ne souciait guère des droits humains. Mais là s'en était trop ! Sans trop de difficultés, elle parvient à passer toute la sécurité du système et à vous libérer de votre cellule.

Malheureusement, le laboratoire est hautement sécurisé et il vous faut encore sortir du bâtiment pour vous échapper. Pas de panique, Clothilde devrait pouvoir vous aider à distance, à moins que ses compétences en cybersécurité ne se soient un peu rouillées à force de nettoyer la salle de serveurs du laboratoire.

Vous et votre camarade recherchez la sortie du laboratoire afin de gagner votre liberté. Croyez-nous, ce ne sera pas une mince affaire, ce laboratoire regorge de portes et chaque porte est verrouillée par une énigme, car cela « stimule la créativité des employés » et les monstres issus d'expériences ratés que Clothilde a malencontreusement libérés en tentant de vous aider ne vont pas vous simplifier la tâche... Saurez-vous vous évader de ce laboratoire infernal ?... Dépêchez-vous ! Le temps presse, les scientifiques peuvent revenir d'un instant à l'autre !

### 1.2.2 Conclusion de l'histoire

Vous parvenez enfin à vous échapper, mais lorsque vous poussez la porte de la sortie vous ne trouvez pas la lumière puissante et la douce chaleur du soleil qu'il vous a semblé percevoir par les fenêtres du labo mais bien la lumière blafarde d'une salle éclairée aux néons. Vous êtes dans une nouvelle salle de laboratoire carrelé de carreaux blancs sans vie qui vous oppresse, seulement, vous n'êtes pas seul... Clothilde est là ! Vous vous sentez tout de suite rassurés. Une sorte de vague de chaleur et de bien-être vous envahit. « Ramenez-les à leurs cellules. » Quatre gardes que vous n'aviez pas vu jusqu'à présent s'avancent et vous saisissent vous et votre compagnon par les bras. Vous vous laissez faire, trop choqué pour réagir...

Clothilde qui vous a pourtant aidé à vous échapper ordonne maintenant votre retour à l'incarcération. On vous retourne face à la grande porte que vous venez de prendre, vous apercevez deux écran numérique affichant respectivement « Crésus : Clone n°397 » et « Crésus : Clone n°398 » ainsi que deux notes très détaillées. Au-dessus de la porte se trouve le logo de Xenor, en dessous un slogan : « Avec Xenor, négationnez la mort ». Vous comprenez alors que Clothilde ne vous aidait pas à vous échapper mais évaluait simplement vos capacités.

En réalité, il y a quelques années de cela, le laboratoire Xenor avait réussi à comprendre où se situe la conscience de l'Homme et à la déplacer d'un corps vers un autre, proposant ainsi aux hommes fortunés de pourvoir s'offrir une seconde jeunesse. La note que vous avez vu sur l'écran ne sert en fait qu'à définir quel clone a les meilleures capacités pour définir lequel accueillera la conscience du client.

### 1.2.3 Intentions/Visée de réflexion du scénario

Notre but ici est de faire réagir, de choquer le joueur pour le faire réfléchir. Que ce soit par la croyance en une vie après la mort, une réincarnation ou autre, ou bien par la science, l'Homme cherche depuis toujours à trouver une solution à sa finitude. En effet, c'est bien là la vocation de la médecine : de repousser cette échéance qui caractérise l'être humain ; tout faire pour éviter notre fin ou en tout cas de la repousser au maximum.

Et on voit d'ailleurs chaque jours ses succès ! Prenons un exemple qui peut à première vue sembler anodin : si vous êtes né par césarienne, il y a quelques dizaines d'années, vous et votre mère seraient inévitablement morts. De nombreuses maladies autrefois mortelles sont désormais sans importance. On peut même guérir d'un cancer<sup>1</sup> s'il est détecté assez tôt !

Et pour repousser sa fin, l'Homme est prêt à tout, car le temps est sa ressource la plus chère car fatidiquement limitée.

Depuis quelques siècles maintenant, les philosophes tentent de caractériser notre conscience. Les scientifiques tentent eux de comprendre ce qu'elle est et où elle est située dans le cerveau. Sommes-nous ancrés à notre corps ou notre âme est-elle volatile ? Poumons-nous comme le laisse entendre certains

1. <https://www.cancer.be/le-cancer/de-la-r-mission-la-gu-rison/gu-rison-du-cancer>

sortir de notre enveloppe charnelle ?...

Imaginez maintenant que l'on réussisse à définir, situer et transplanter notre conscience dans un autre corps. Ce serait là, la fin de la finitude. Une grande question morale apparaîtrait alors sur le droit moral de l'Homme à transplanter sa conscience dans le corps d'un autre.

De la même façon que les premières chirurgies, les premières greffes ont soulevée de nombreuses questions morales sur leur bien-fondé. La différence réside ici dans le consentement. Mais lorsque l'on parle de réaliser des greffes de porc sur l'homme, il n'y a pas de notion de consentement. Lors des essais cliniques, on ne demande pas non plus leur avis aux souris.

Le but n'est pas de remettre en cause toutes ces pratiques, loin de là, mais plutôt de se questionner sur les limites. Jusqu'où pouvons-nous aller ? Est-ce que la vie éternelle est une raison suffisante pour « cultiver » des êtres humains pour y implanter sa conscience ?

Mais une chose peut être sûre, si cela s'avère possible, et même si c'est interdit, les plus fortunés pourront se l'offrir. Il existe bien un trafic d'organes, un trafic d'esclaves, alors pourquoi pas un trafic d'humain à destination de transplantation de conscience ?

Transplanté sa conscience dans un autre pourrait devenir au fil des ans aussi banal que de prendre un médicament...

Et si l'on créait un clone de nous-même pour y implanter sa conscience ?... Serait-ce plus moral ?

Nous voulons faire réfléchir sur ce qu'impliquerait la vie éternelle par transplantation de conscience et sur le bien-fondé de notre volonté de combattre notre nature en repoussant toujours plus les limites de la mort... Où doit-on s'arrêter ?

## 1.3 Mécaniques de GamePlay

### 1.3.1 Présentation Générale

Heureusement pour vous vous n'êtes pas seul, votre compagnon de cellule vous accompagne et croyez-nous, la coopération sera maître mot pour espérer sortir un jour de ce laboratoire infernal. Il vous faudra résoudre les énigmes pour accéder ou déverrouiller les portes.

Dans certaines salles, se trouveront des monstres, vous disposerez de fioles pour les combattre. Ils pourront également être combattu à la main, au corps à corps. Vous aurez également une barre de vie car les monstres ne sont pas inoffensifs. Dans le cas où l'un des deux joueurs meurt pendant un affrontement avec un monstre, où d'une quelconque autre manière, le deuxième joueur pourra réanimer le premier grâce au super kit de secours. Seulement, si les deux joueurs viennent à être éliminés par le monstre, vous avez perdu car les scientifiques vous retrouveront et n'aurons aucun intérêt à vous réanimer, la Terre grouille de cobayes...

Les scientifiques développent également des boosts de jeunesse, il y en a quelques-uns qui traînent dans le laboratoire, ils devraient vous permettre de récupérer de la vie si vous avez l'oeil.

Les joueurs seront toujours dans la même pièce. En effet, prendre une porte emmène votre camarade de cellules avec vous, vous ne voudriez pas vous retrouver seuls dans ce laboratoire...

Les énigmes sont conçues pour être résolues à deux.

Le jeu comportera un lobby principal reliant les niveaux entre eux. Ceux-ci peuvent prendre plusieurs formes : énigmes et/ou combats contre des monstres.

Certains niveaux seront générés procéduralement pour augmenter la rejouabilité.

Chaque joueur contrôlera un personnage dont les caractéristiques sont :

- une barre de vie
- un inventaire
- possibilité de déplacer
- possibilité d'attaquer

- possibilité d'interagir avec certains éléments du décors
- possibilité de collecter certains objets
- possibilité de d'utiliser certains objets collectés
- possibilité de se soigner
- possibilité de réanimer son coéquipier
- menu pause et configurations

### 1.3.2 Les Énigmes

#### Labyrinthe à Boîtes

But : Bouger des boîtes pour se frayer un chemin jusqu'à la sortie.

Les joueurs apparaissent à deux endroits différents dans la salle.

La salle est pleine de boîte de rangement (nous sommes dans un entrepôt du laboratoire).

Les joueurs peuvent interagir avec les boîtes (les tirer, les pousser, les décaler sur les côtés).

Leurs chemins jusqu'à la sortie sont bloqués par des boîtes mais ils ne pourront pas toujours se débloquer seuls, ils auront besoin de l'aide de leur coéquipier pour bouger certaines boîtes qui étaient jusqu'alors impossible à bouger pour eux.

Certaines boîtes sont trop lourdes pour être poussées.

Pour sortir de la salle, il faut que chaque joueur se retrouve sur une plaque de pression, en face d'une porte.

Il n'y a pas de condition d'échec sur cette énigme.

#### Labyrinthe Invisible

But : Sortir du labyrinthe invisible en coopérant avec votre coéquipier.

Un joueur aura la carte du labyrinthe et l'autre sera dans le labyrinthe, mais le labyrinthe est invisible. En effet, les murs de ce labyrinthe sont invisibles.

Marcher à un endroit interdit inflige des dégâts et téléporte le joueur au début du labyrinthe.

Si le joueur meurt, c'est une condition d'échec.

#### Enigmes des fils

But : Couper le bon fil à partir de consignes, un joueur ne peut pas couper les fils et avoir les consignes en même temps.

Ce système est inspiré du jeu « Keep Talking and Nobody Explode ».

Un joueur sera face aux fils, l'autre aura un manuel. Le manuel contiendra des phrases possédantes chacune une condition et une conséquence (ex : « si le premier fil est bleu couper le dernier fil »).

Si l'un des joueurs coupe un mauvais fil, il perd de la vie en laissant une limite de vie d'un demi cœur au joueur, l'empêchant ainsi de mourir sur cette énigme.

### 1.3.3 Les Monstres

Les monstres seront des ennemis se situant sur notre chemin un peu partout dans les niveaux. Ils auront tous un lien avec leur environnement : le laboratoire. La taille, force, vie des monstres ainsi que leurs attaques varient d'un monstre à l'autre et d'un endroit à un autre.

Les différents monstres sont :

**Slime** : Un des vestiges des expériences précédentes des scientifiques.

Ils peuvent être de différentes tailles, cela influe sur ses statistiques.

Ce monstre combat au corps-à-corps.

**Robots :** Ils sont chargés de protéger le laboratoire et d'éliminer les intrus tels que vous.

Il y a plusieurs types de robots, de différentes formes.

Leurs attaques varient, certains attaquent au corps-à-corps, d'autres à distance.

#### 1.3.4 Les Fioles

Les fioles sont le fruit de plusieurs années d'expériences menées par les scientifiques de ce laboratoire. Elles donnent plusieurs effets : des bonus comme un regain de vie par exemple, mais certains ratés peuvent également donner un malus au joueur comme la perte de PV.

Les effets recensés par les scientifiques sont les suivants :

- Effet de soin
- Augmentation de la force
- Effet de dégâts

A cause de ce large nombre d'effets négatifs, la production de ces fioles ont cessé. De plus, certains effets n'ont pas encore été découvert par les scientifiques.

#### 1.3.5 Labyrinthe fléché

Cette partie a été réalisée par Hugo.

Ce labyrinthe est constitué de dalles fléchées, le joueur ne peut aller que dans la direction des flèches et ne peut pas aller en opposition à une flèche sur une autre tuile.

De plus les deux joueurs ne verront pas toutes des dalles les obligeant ainsi à communiquer pour pouvoir résoudre ce labyrinthe. (chacun voit la moitié des dalles)

Si un joueur ne respecte pas les règles précédemment citées, il perdra de la vie, et sera possiblement téléporté au début du labyrinthe.

## Chapitre 2

# Rappel Des Objectifs pour la Soutenance 2

TABLE 2.1 – Répartition des Responsabilités

Tâches	Elya	Ethan	Hugo	Timothy
Création du Lobby		R		
Intégration des premières énigmes dans le jeu			R	S
Création des premiers monstres.	R			S
Création du système de Narration.		R		
Implémentation de l'énigme du labyrinthe à boîtes.		R		
Début de création du texte de la narration.		R		
Début de création du site (FrontEnd)		R		
Début de création du site (BackEnd)			R	
Création des assets principaux (sols, murs, portes)	R			
Début création du launcher			R	
Début création système multijoueur			S	R
Début recherche musique				R

Légende :

Symbol	Signification
Responsable	R
Suppléant	S

# Chapitre 3

## Avancement

Durant cette période, nous avons travaillé chacun de notre côté en se répartissant les tâches.

### 3.1 Graphismes

Nous avons beaucoup avancé sur cette partie, les assets<sup>1</sup> pour les murs ainsi que le sol et les portes ont été créés.

Nous nous contenterons ici de présenter uniquement le visuel des assets, leurs utilités/fonctionnalités étant expliquées dans la suite du rapport.

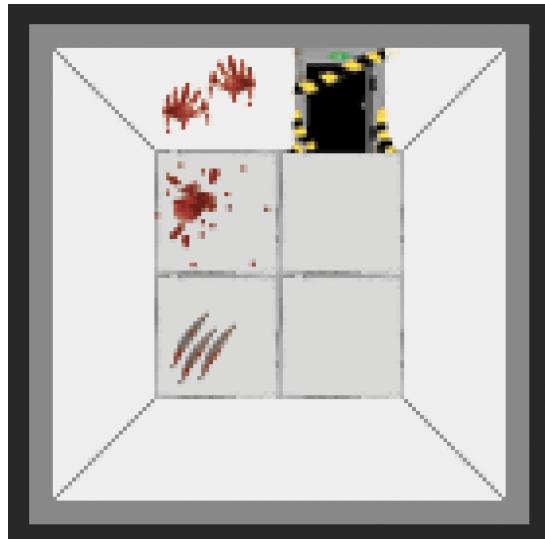


FIGURE 3.1 – Exemple de salle 1

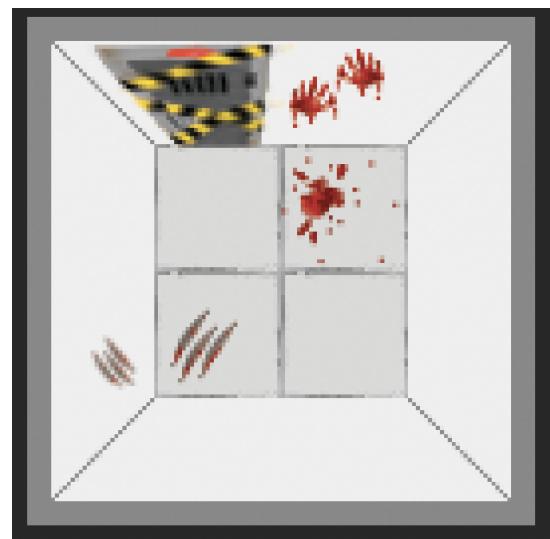


FIGURE 3.2 – Exemple de salle 2

---

1. Dans ce contexte : Représente un élément d'une planche de plusieurs dessins



FIGURE 3.3 – Exemple de salle 3



FIGURE 3.4 – Exemple de porte

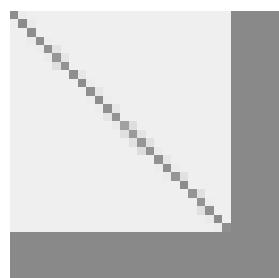


FIGURE 3.5 – Asset de mur 1

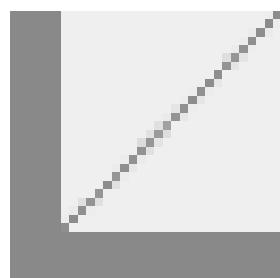


FIGURE 3.6 – Asset de mur 2

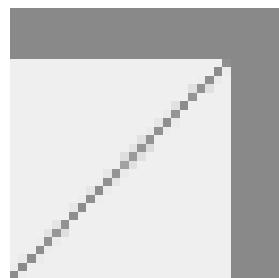


FIGURE 3.7 – Asset de mur 3

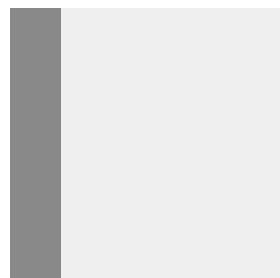


FIGURE 3.8 – Asset de mur 4

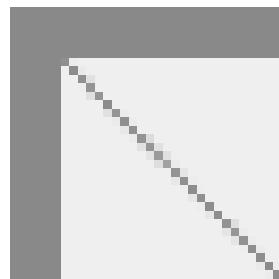


FIGURE 3.9 – Asset de mur 5



FIGURE 3.10 – Asset de mur 6

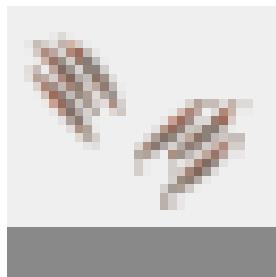


FIGURE 3.11 – Asset de mur griffes 1

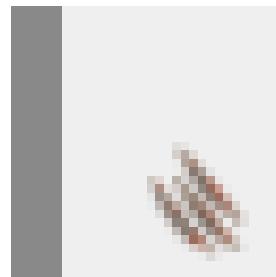


FIGURE 3.12 – Asset de mur griffes 2

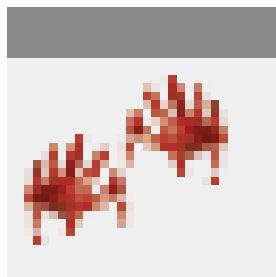


FIGURE 3.13 – Asset de mur sang 1



FIGURE 3.14 – Asset de mur sang 2

Les assets des portes sont purement décoratifs, on rappelle qu'il faut les plaques de pressions pour pouvoir changer de salle.

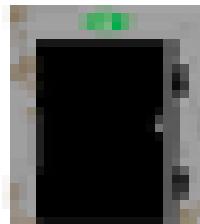


FIGURE 3.15 – Asset de porte 1

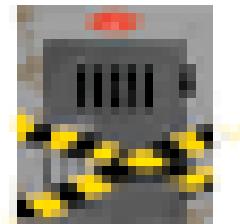


FIGURE 3.16 – Asset de porte 2



FIGURE 3.17 – Asset de porte 3

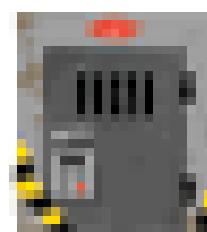


FIGURE 3.18 – Asset de porte 4

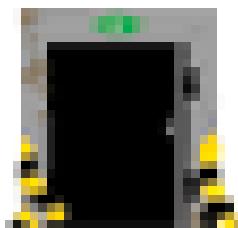


FIGURE 3.19 – Asset de porte 5

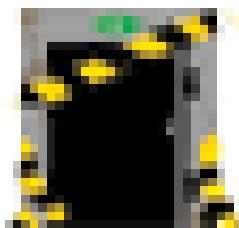


FIGURE 3.20 – Asset de porte 6

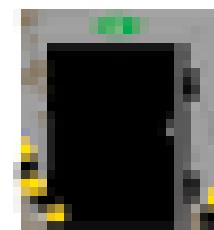


FIGURE 3.21 – Asset de porte 7



FIGURE 3.22 – Asset de porte 8

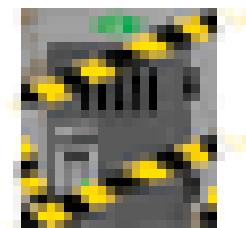


FIGURE 3.23 – Asset de porte 9

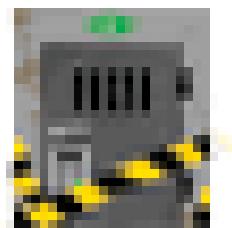


FIGURE 3.24 – Asset de porte 10

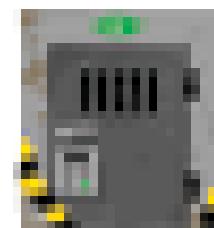


FIGURE 3.25 – Asset de porte 11

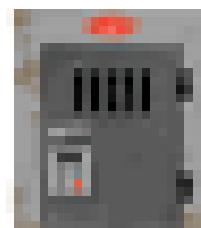


FIGURE 3.26 – Asset de porte 12



FIGURE 3.27 – Asset de sol



FIGURE 3.28 – Asset de sol griffes



FIGURE 3.29 – Asset de sol sang



FIGURE 3.30 – Asset de sol slime

## 3.2 Les Personnages

Cette partie a été réalisé par Elya. Afin de résoudre les problèmes de la soutenance 1 (couleur différente selon l'animation du personnage, ombres qui apparaissaient et disparaissaient), de nouvelles planches ont été créés ainsi que de nouveaux assets pour la mort du personnage et l'animation qui lui est associée.

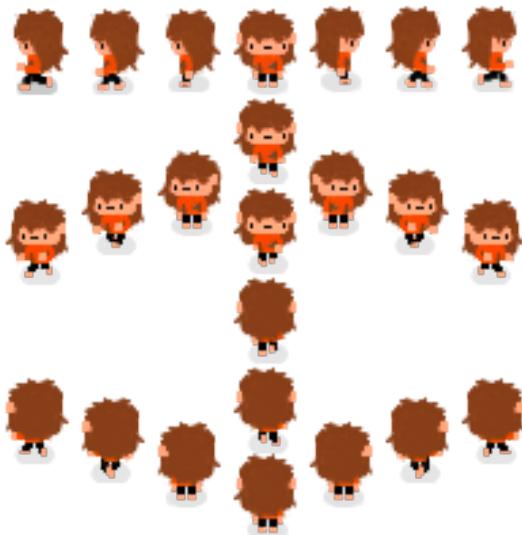


FIGURE 3.31 – Player Femme



FIGURE 3.32 – Player Homme

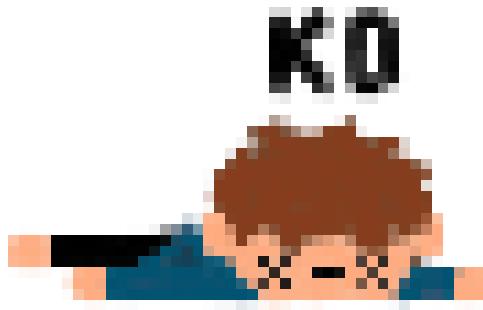


FIGURE 3.33 – Player K.O. Homme

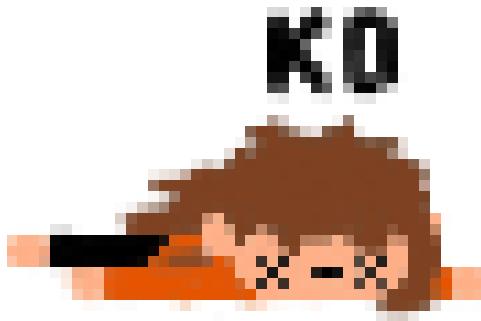


FIGURE 3.34 – Player K.O. Femme

De plus, dans la classe player nous avons ajouté l'attribut `strength` qui définit la force d'attaque du joueur.

#### Listing 3.1 – Script de la classe Player

```
1 public class Player : MonoBehaviour
2 {
3
4     //Attack of the player
5     public int Strength = 15;
6
7 }
```

### 3.3 Création des Salles Communes

Les éléments de cette section ont été réalisé par Ethan.

#### 3.3.1 Les Cellules

Les cellules sont les salles de début du jeu, elles ont été complètement réalisé (hormis le dialogue qui n'est pas le bon). Dans la salle est appliquée le GameManager qui ne sera jamais détruit au chargement de nouvelles scènes, le fait que l'on ne soit dans les cellules qu'au début du jeu permet de ne pas dupliquer l'objet. La salle a aussi un intérêt narratif car elle plonge le joueur dans l'histoire.

J'ai réalisé trois nouveaux assets pour cette salle (pour le mur de verre au milieu de la scène), ils sont visibles sur le screen ci-dessous de la salle

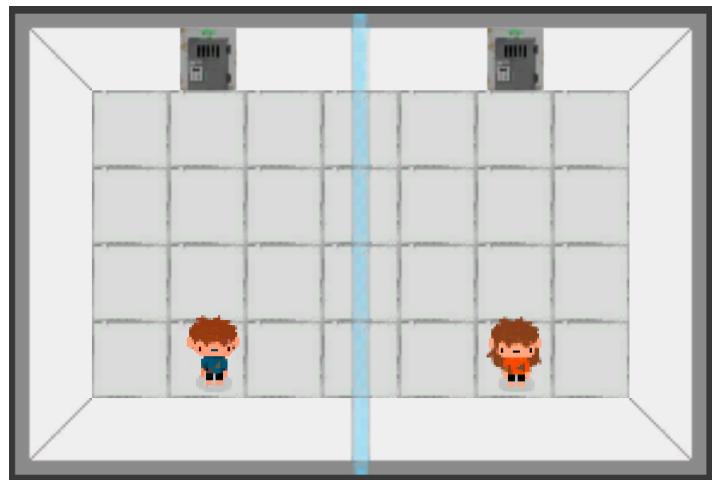


FIGURE 3.35 – Les cellules

#### 3.3.2 La Salle Principale

La salle principale est la salle où réapparaîtront les joueurs à la fin des énigmes, elle permet d'accéder au différents niveaux et à la porte de sortie.

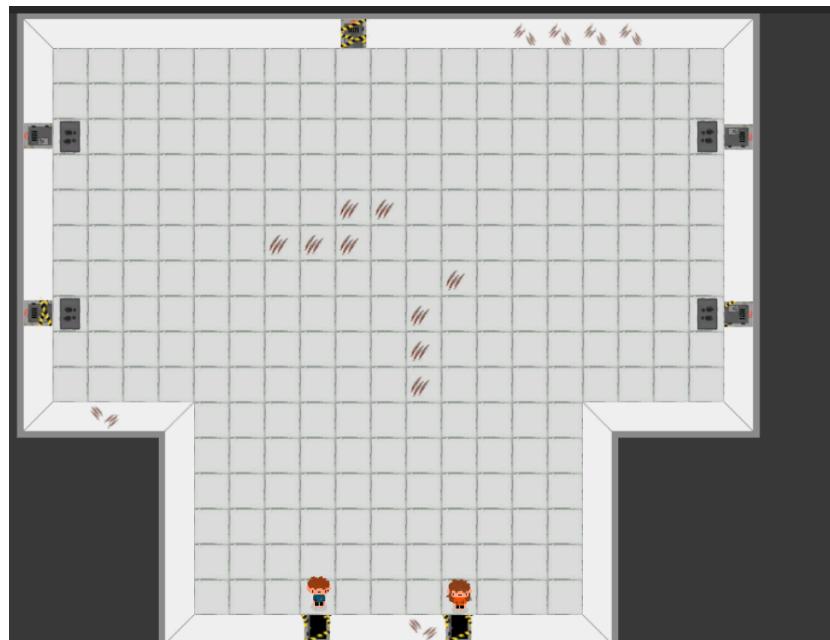


FIGURE 3.36 – La Salle Principale

### 3.4 Les monstres

Cette partie a été réalisé par Elya.  
Pour les monstres, nous avons dessiné différents assets selon les types de monstres et leurs attaques.  
D'abord nous avons les slimes verts et rouges.



FIGURE 3.37 – Monstre Slime Vert



FIGURE 3.38 – Monstre Slime Rouge



FIGURE 3.39 – Attaque Monstre Slime Vert 1



FIGURE 3.40 – Attaque Monstre Slime Vert 2



FIGURE 3.41 – Attaque Monstre Slime Vert 3

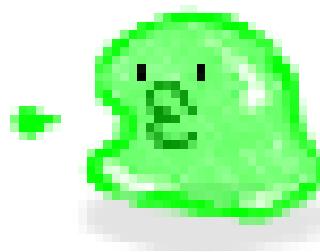


FIGURE 3.42 – Attaque Monstre Slime Vert 4

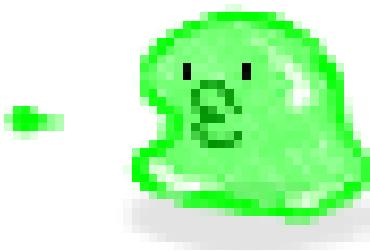


FIGURE 3.43 – Attaque Monstre Slime Vert 5



FIGURE 3.44 – Attaque Monstre Slime Vert 6



FIGURE 3.45 – Attaque Monstre Slime Vert 7

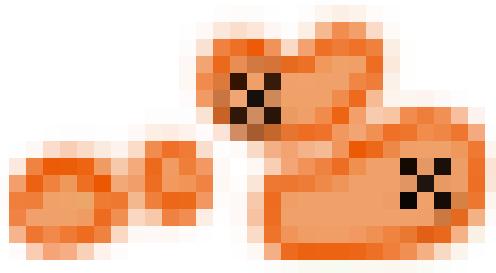
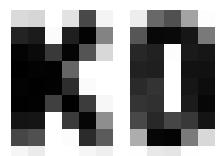


FIGURE 3.46 – Mort Monstre Slime 1

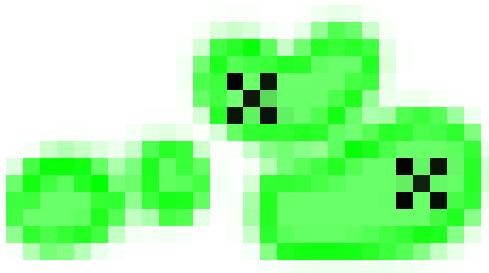
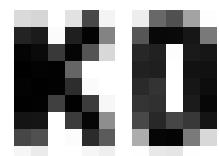


FIGURE 3.47 – Mort Monstre Slime 2

Ensuite nous avons quatre formes de robots différentes.

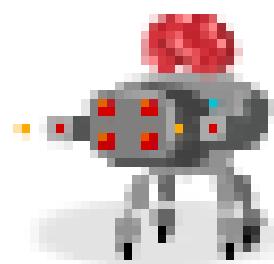


FIGURE 3.48 – Monstre Robot 1

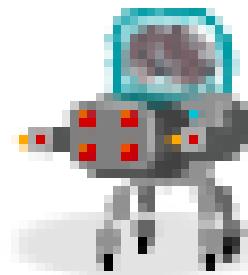


FIGURE 3.49 – Monstre Robot 2

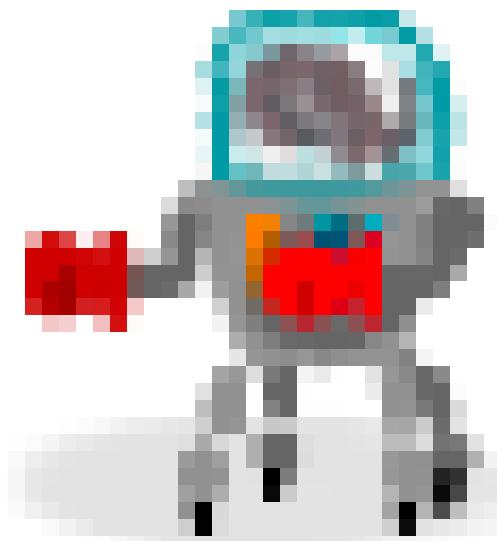


FIGURE 3.50 – Monstre Robot 3

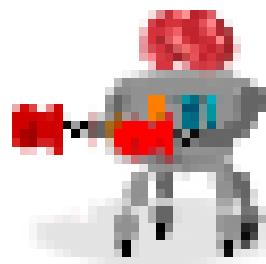


FIGURE 3.51 – Monstre Robot 4

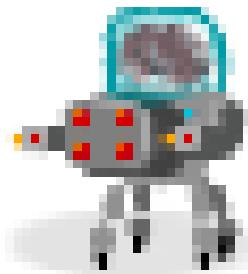


FIGURE 3.52 – Attaque Monstre Robot 1

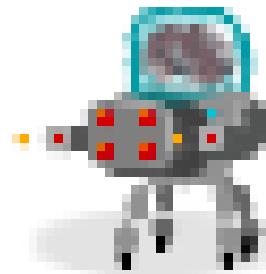


FIGURE 3.53 – Attaque Monstre Robot 2

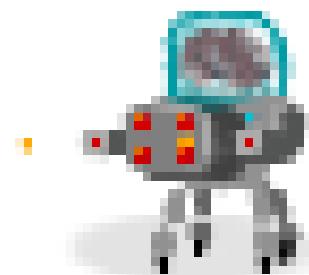


FIGURE 3.54 – Attaque Monstre Robot 3

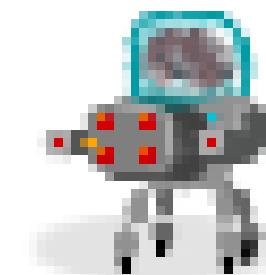


FIGURE 3.55 – Attaque Monstre Robot 4

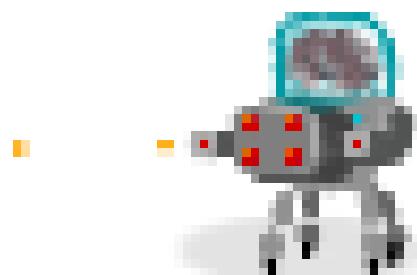


FIGURE 3.56 – Attaque Monstre Robot 5

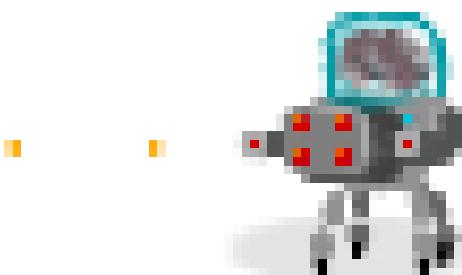


FIGURE 3.57 – Attaque Monstre Robot 6

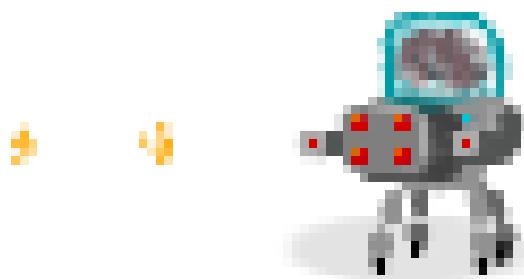


FIGURE 3.58 – Attaque Monstre Robot 7

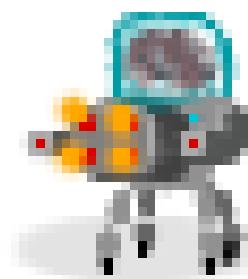


FIGURE 3.59 – Attaque Monstre Robot 8

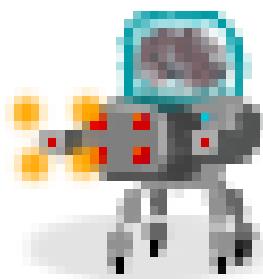


FIGURE 3.60 – Attaque Monstre Robot 9

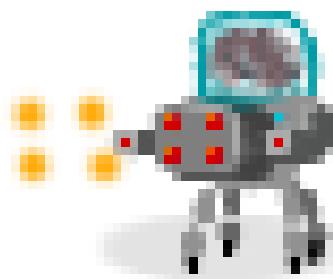


FIGURE 3.61 – Attaque Monstre Robot 10

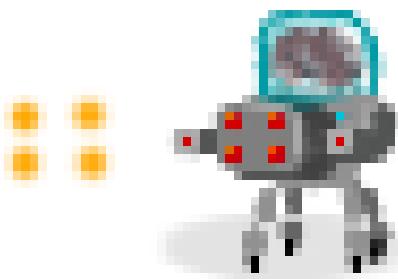


FIGURE 3.62 – Attaque Monstre Robot 11

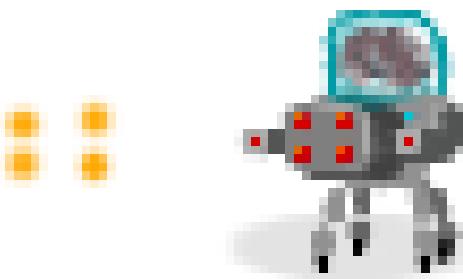


FIGURE 3.63 – Attaque Monstre Robot 12

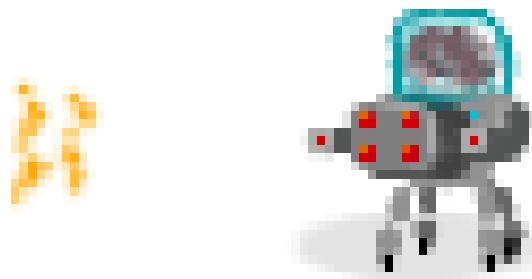


FIGURE 3.64 – Attaque Monstre Robot 13

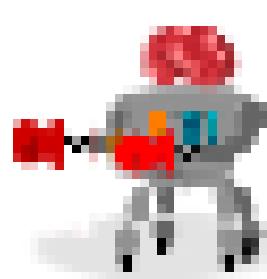


FIGURE 3.65 – Attaque Monstre Robot 14

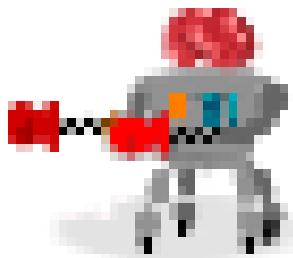


FIGURE 3.66 – Attaque Monstre Robot 15

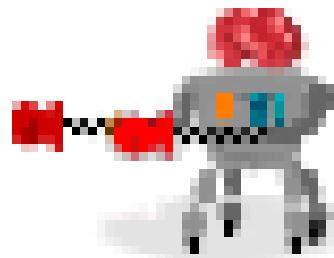


FIGURE 3.67 – Attaque Monstre Robot 16

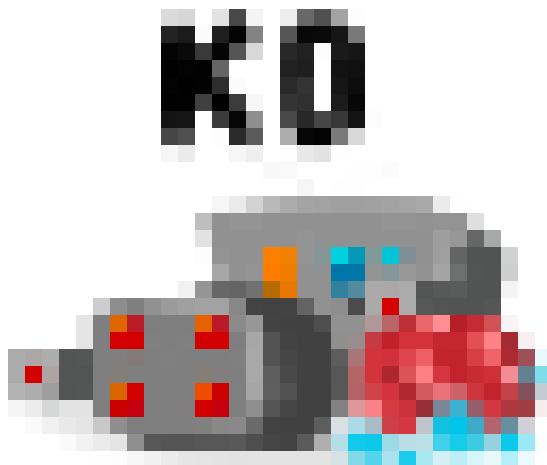


FIGURE 3.68 – Mort Monstre Robot 1

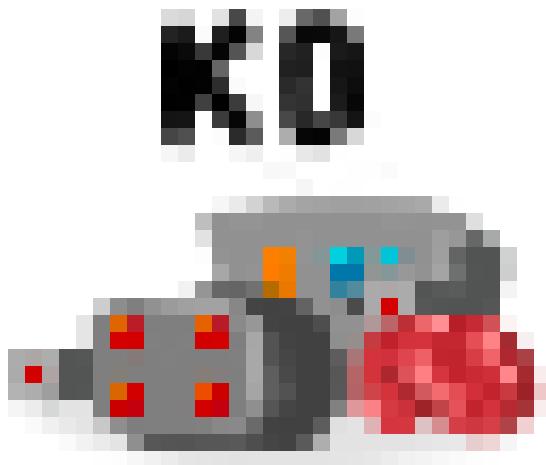


FIGURE 3.69 – Mort Monstre Robot 2

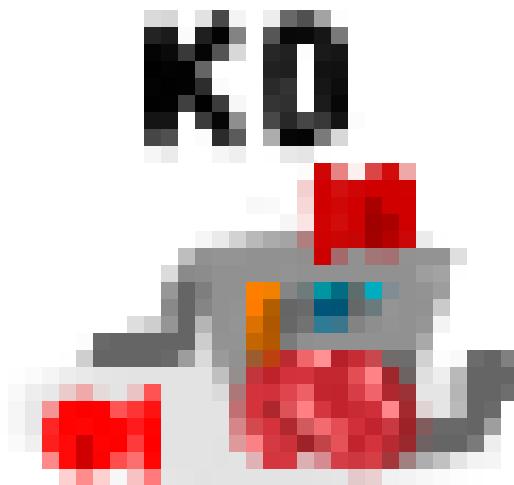


FIGURE 3.70 – Mort Monstre Robot 3

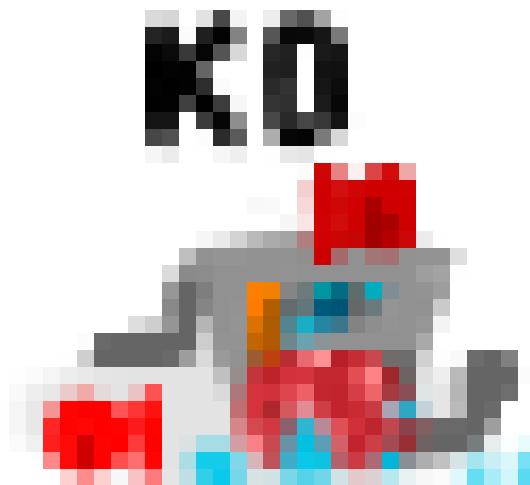


FIGURE 3.71 – Mort Monstre Robot 4

De plus, on associe deux classes aux monstres :

**Listing 3.2 – Script de la classe Monstre**

```

1 public class Monsters : MonoBehaviour
2 {
3     Transform target;           // Target of Pathfinding
4     NavMeshAgent agent;        // Pathfinding
5     float[] distanceFactor;   // Facteur de distance entre les joueurs < 1 (ex :
6         1/2 -> change de target si l'autre joueur est à la moitié de la distance de
7         l'autre)
8     GameObject[] playersObject;
9     Transform[] playersTransform;
10    public Animator animator;
11    int round;
12
13    // Stats of monster
14    public float attack;
15    float pv;
16    float start_pv;
17
18    // Make AI do things
19    public void AI()
20    {
21        CheckDistance();      // Check if should change target
22
23        if (!agent.isStopped) // If close to target
24        {
25            agent.SetDestination(target.position); // Move GameObject
26            animator.SetFloat("Moving", (0.5f)); // Moving animation
27        }
28        else
29        {
30            Attack();
31        }
32
33        // Check if should change target if one player is closer than the other
34        void CheckDistance()
35        {
36            Transform tmp = target; // Original target
37            float[] distance = new float[2];
38
39            // Get distance to players
40            for (int i = 0; i < playersTransform.Length; i++)
41            {
42                target = playersTransform[i];
43                distance[i] = agent.remainingDistance;
44            }
45
46            float factor = 1;
47
48            // Get distance factor between players
49            if (distance[1] != 0)
50                factor = distance[0] / distance[1];
51
52            // Check if should change target according to players position and given
53            // distance factor
54            if (factor < distanceFactor[0])
55                target = playersTransform[0]; // Change target
56            else if (factor > 1 / distanceFactor[1])
57                target = playersTransform[1]; // Change target
58            else
59                target = tmp; // Reset target
60
61        public void GetDamage(float damage)
62        {
63            pv -= damage;
64
65            if (pv <= 0)
66                Die();
67            else
68            {
69                // Run damaged animation

```

```

69         animator.SetFloat("Damage", (0.5f));
70     }
71 }
72
73 void Die()
74 {
75     // Run death animation
76     animator.SetFloat("Die", 1);
77 }
78
79 void Attack()
80 {
81     // Run Attack animation
82     if (round % 2 == 0)
83     {
84         animator.SetFloat("Attack", (0.5f)); //Animation attack number 1
85     }
86     else
87     {
88         animator.SetFloat("Attack", (0.1f)); //Animation attack number 2
89     }
90 }
91
92 public void Heal()
93 {
94     pv = start_pv;
95 }
96 }
```

Cette classe définit ce que le monstre peut faire (mourir, attaquer, détecter les joueurs proches, guérir et l'IA), met en place les animations selon l'action effectuée et les attribut qui lui sont associés (la vie du monstre sous le nom de pv et la force de son attaque).

**Listing 3.3 – Script de la mise en place de l'IA des monstres**

```

1 public class SampleEnemy : MonoBehaviour
2 {
3     Monsters monstre;
4     // Start is called before the first frame update
5     void Start()
6     {
7         monstre = new Monsters(1, 1, 1, 0.1f, 1 / 4); // Pif values
8     }
9
10    // Update is called once per frame
11    void FixedUpdate()
12    {
13        monstre.AI();
14    }
15 }
```

Cette classe met en place la détection des joueurs pour pouvoir ensuite lancer l'IA des monstres.

### 3.5 Potions

Cette partie a été réalisé par Elya.



FIGURE 3.72 – Potion Verte 1



FIGURE 3.73 – Potion Verte 2



FIGURE 3.74 – Potion Verte 3



FIGURE 3.75 – Potion Rose 1



FIGURE 3.76 – Potion Rose 2



FIGURE 3.77 – Potion Rose 3

Ce sont ces assets qui permettent de créer l'animation de la potion et l'Animator<sup>2</sup> permet de la mettre en place en boucle en permanence.

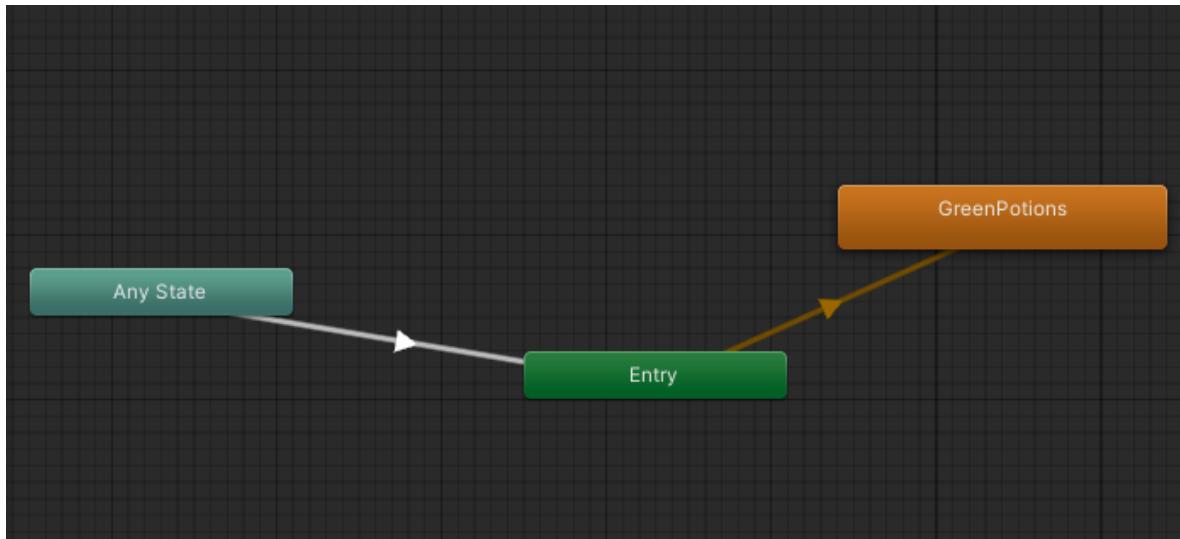


FIGURE 3.78 – Animator Potion Verte

2. Animator : outil d'Unity qui permet le lancement d'animation selon une ou plusieurs variables

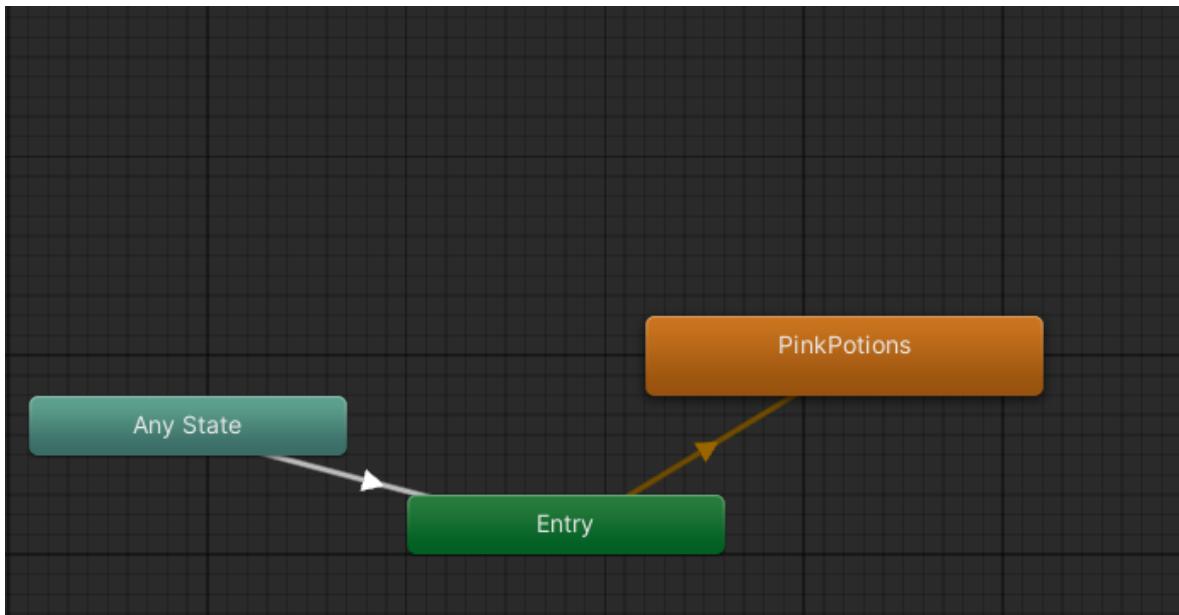


FIGURE 3.79 – Animator Potion Rose

Listing 3.4 – Script de la classe Potions

```

1
2 public class Potion : MonoBehaviour
3 {
4
5
6     public life vie;//the player that fight
7     public player joueur;
8     public Monsters monstre;
9
10    /*
11     * Add Object hitbox in children of Potion
12     */
13
14
15     //Type of differents potions
16     enum Type
17     {
18         Damage ,
19         Heal ,
20         Stregth
21     }
22
23     Type type;
24
25     //initiate the value of their type
26     float damage = 0;
27     float heal = 0;
28     float streng = 0;
29
30     Potion(Type type , float x)
31     {
32         this.type = type;
33
34         switch (type)
35         {
36             case Type.Damage:
37                 damage = x;
38                 break;
39             case Type.Heal:
40                 heal = x;
41                 break;
42             case Type.Stregth:
43                 streng= x;
44                 break;
45             default:
46                 break;
47         }
48     }
49
50     // Make effect of potion when used
51     public void Effect(Collider2D obj)
52     {
53         switch (type)
54         {
55             case Type.Damage:
56                 if (obj.tag == "Player")
57                 {
58                     vie.Reduce4(1);//A function of the life class that reduce
59                     by 1/4 the life of the player
60                 }
61                 else
62                 {
63                     monstre.GetDamage(15);//A function of the Monsters class
64                     that reduce the life of the monster
65                 }
66                 break;
67             case Type.Heal:
68                 if (obj.tag == "Player")
69                 {

```

```

68                     vie.HealMax(); //A function of the life class that goes up
69                     to the max the life of the player
70                 }
71             else
72             {
73                 monstre.Heal(); //A function of the Monsters class that
74                 goes up to the max the life of the monster
75             }
76             break;
77         case Type.Strength:
78             if (obj.tag == "Player")
79             {
80                 joueur.Strength += 5; //Add more strength to the player
81             }
82             else
83             {
84                 monstre.attack += 3; //Add more strength to the monster
85             }
86             break;
87         default:
88             break;
89     }
}
}

```

Ce script représente la classe Potion. On peut voir qu'il y a trois type de potions : une qui guérit (Heal), une qui retire de la vie (Damage) et une qui ajoute de la force (Strength).

Les Potions peuvent agir aussi bien sur les monstres que sur les joueurs.

Selon le type de la potion, la fonction Effect va effecter l'effet de la potion.

### 3.6 Menu Principal

Cette partie a été réalisé par Timothy.  
Le menu a été créé à l'aide de l'asset « Simple Menu » provenant du Unity Asset Store<sup>3</sup>. Il possède plusieurs fenêtres liées entre-elles.

- Un menu « Principal » avec les options Play (fonctionnel) amenant au menu « Jouer », Settings (fonctionnel) amenant au menu « Paramètre », et Exit (fonctionnel), faisant quitter le jeu.



FIGURE 3.80 – Menu Principal

- Un menu « Paramètre » où l'on peut ajuster le volume du jeu (non fonctionnel), activer/désactiver le mode plein écran (fonctionnel), changer les qualité visuelle du jeu (fonctionnel).

3. Unity Asset Store est un marketplace d'assets pour Unity créés par Unity et sa communauté

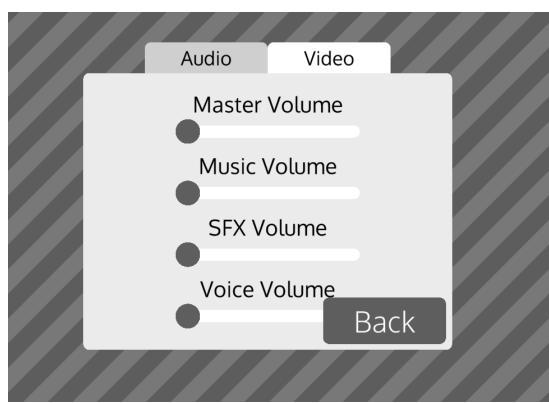


FIGURE 3.81 – Menu Audio

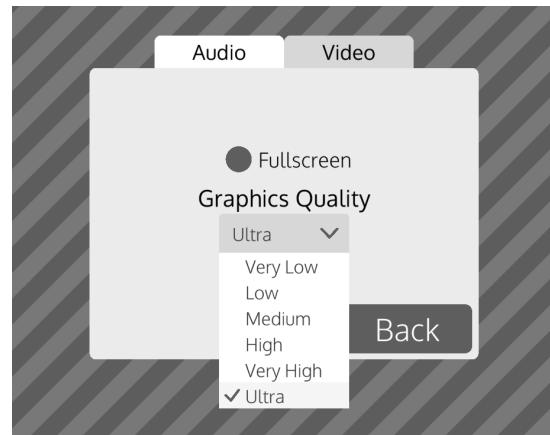


FIGURE 3.82 – Menu Video

- Un menu « Jouer » où l'on peut choisir de continuer une partie précédente ou créer une nouvelle partie. Ces deux options amènent au menu « En ligne » et il n'y a pour l'instant aucune différence entre les deux.



FIGURE 3.83 – Menu Principal

- Un menu « En Ligne » où l'on peut créer ou rejoindre une partie en ligne (fonctionnel).



FIGURE 3.84 – Menu En Ligne

Le design du menu est susceptible de changer.

## 3.7 Multijoueur en ligne

Le multijoueur a été implémenté dans Unity grâce à Photon PUN<sup>4</sup>. Photon fonctionne grâce à un système de Room<sup>5</sup> que les joueurs peuvent créer et rejoindre afin de jouer en ligne.

### 3.7.1 Implémentation dans Unity

Par défaut tous les objets présents dans une scène sont intanciés en local, si un joueur les modifie en jeu, les autres ne le verront pas dans le jeu.

Pour instancier des objets sur le serveur ils faut leurs attacher plusieurs scripts fournis par Photon, ainsi que les mettre dans un dossier spécial appelé « Ressources ».

- « Photon View » permet d'intancier des objets sur le serveur.

**Listing 3.5 – Instancie un objet sur le serveur**

```
1 // Instancie un objet sur le serveur
2 PhotonNetwork.Instantiate(string prefabName, Vector3 position, Quaternion rotation)
```

- « Photon Transform View » permet de synchroniser la position, rotation et l'échelle de taille d'un objet.
- « Photon Animator View » permet de synchroniser les animations des objets.

## 3.8 Énigme : Le Labyrinthe de Boîtes

Les avancements sur le labyrinthe de boîtes ont été réalisés dans leurs intégralités par Ethan.

### 3.8.1 Création des Objets pour l'Enigme

Tous les objets nécessaires à cette énigme ont été créés.

#### Boite Immobile

La boîte immobile, à l'aide d'un simple BoxCollider2D<sup>6</sup>, elle empêche le joueur de passer sur la boîte (l'objet bloque son avancement, forçant ainsi le joueur à le contourner).

#### Boite qui bouge

La boîte qui bouge a deux fonctionnalités, elle doit pouvoir être poussée si le joueur marche dans sa direction et tirée si le joueur appuie sur une touche pour la tirer.

Pour pousser la boîte, j'ai utilisé un simple BoxCollider2D un Rigidbody2D<sup>7</sup>

Pour tirer la boîte, j'ai eu besoin d'un autre BoxCollider2D (légerement plus grand) qui, lorsque le joueur entre en collision, active un script qui permet de lier la boîte au player s'il appuie la touche demandée. La liaison est réalisée avec un FixedJoint2D<sup>8</sup> et d'un script qui lie/délie la boîte

---

4. Photon PUN est une réimplémentation et amélioration du système de multijoueur en ligne de Unity

5. Les Rooms sont équivalents à des sessions de jeu en ligne

6. Composant de Unity qui permet de gérer les collisions avec des Objets

7. Autre composant de Unity qui permet de gérer la position d'un objet en fonction des forces qui s'appliquent sur lui

8. Également un élément de Unity qui permet de lier entre eux, deux objets et donc de lier leur déplacement, boîtes de collisions, etc.

Listing 3.6 – Script de la boîte qui bouge

```

1 //Appelée à chaque frame
2 private void Update()
3 {
4     //check si on appuie sur E ssi en contact avec la boîte est pressée
5     if (Input.GetKeyDown(KeyCode.E))
6     {
7         //si qqn entre en contact avec la boîte ET que la boîte n'est pas déjà liée
8         //à qqn
9         if (pressed && !linkedToPlayer)
10        {
11            // on la lie au perso
12            Link();
13        }
14        //sinon, si la boîte est déjà liée à ce qqn, on lui fais l'cher la boîte
15        else if (linkedToPlayer)
16        {
17            UnLink();
18        }
19    }
20
21 /**
22 * <summary>Affiche le message pour proposer à l'utilisateur d'intéragir</summary>
23 *
24 * <param name="other">objet avec qui on a collisionné (ici le joueur)</param>
25 *
26 * <returns>Return nothing</returns>
27 */
28 protected override void OnPressure(Collider2D other)
29 {
30     //On affiche le message qui indique au joueur comment intéragir avec la porte.
31     //Grace à fonctionnalité lock le message ne s'affichera que si c pas lock
32     MessageOnScreenCanvas.GetComponent<FixedTextPopUP>().PressToInteractText("Press
33         E to start pulling the box");
34
35 /**
36 * <summary>Lier l'objet avec le premier player rentré en collision avec l'objet</
37 * summary>
38 * <returns>Return nothing</returns>
39 */
40 private void Link()
41 {
42     //le premier joueur qui est entré en collision avec la boîte stocké dans
43     //PressurePlate.cs
44     playerLinked = player[0].gameObject;
45
46     //on récupere le component qui permet de faire le lien
47     FixedJoint2D lienActuel = playerLinked.GetComponent<FixedJoint2D>();
48     //On stocke la position de la boîte et du joueur
49     //Cela permet d'éviter que le point d'ancre fasse bouger la boîte est le
50     //joueur au moment de l'attache
51     Vector2 posBox = this.transform.position;
52     Vector2 posPlayer = playerLinked.transform.position;
53     //On active le component
54     lienActuel.enabled = true;
55     lienActuel.anchor = posBox;
56     lienActuel.connectedAnchor = posPlayer;
57     //On lie au component le rigidbody de la boîte
58     lienActuel.connectedBody = this.GetComponent<Rigidbody2D>();
59
60     //on indique que le lien a été établi avec succ s
61     linkedToPlayer = true;
62 }
63 /**
64 * <summary>Délier l'objet du player avec lequel il était lié</summary>
65 */

```

```

65 * <returns>Return nothing</returns>
66 */
67 private void UnLink()
68 {
69     //on récupere le component qui permet de faire le lien
70     FixedJoint2D lienActuel = playerLinked.GetComponent<FixedJoint2D>();
71     //On le désactive
72     lienActuel.enabled = false;
73     //On réinitialise le lien
74     lienActuel.connectedBody = null;
75
76     //On réinitialise les variables
77     linkedToPlayer = false;
78     playerLinked = null;
79 }

```

Dans ce script, est géré par la fonction `OnPressure()` l'affichage d'un message sur le Canvas dans le cas où l'on entre en contact avec une boîte. Par la fonction `Update()` est gérée la détection des touches pressées et si le script doit lier ou délier la boîte du joueur (en fonction de s'il était déjà lié ou non à la boîte). Ainsi la fonction `Link()` lie le joueur à la boîte et la fonction `Unlink()` le délie.

Un autre script ajoute également au chargement de la scène un élément `FixedJoint2D` aux joueurs pour pouvoir les lier à la boîte.

**Listing 3.7 – Script qui ajoute les liens aux joueurs**

```

1 // Start is called before the first frame update
2 void Start()
3 {
4     // Initialisation de moyen pour lier les boîtes et le joueur lorsque l'on veut
5     // tirer une boîte
6     foreach (GameObject player in GameObject.FindGameObjectsWithTag("Player"))// 
      pour tout les players
7     {
8         // On ajouter un <FixedJoint2D> et on met autoConfigureConnectedAnchor à
9         // false (sinon il lie pas les objets)
10        // Pour lier les obj sur chaque boîte y aura un script qui permettra de
11        // lier la boîte au FixedJoint2D ou de les délier (avec touche E)
12        // Attention : faudra penser à vérifier que y a pas déjà une boîte accrochée
13        // ...
14        player.AddComponent<FixedJoint2D>();
15        player.GetComponent<FixedJoint2D>().enabled = false;
16        player.GetComponent<FixedJoint2D>().autoConfigureConnectedAnchor = false;
17    }
18 }

```

## Le Tabouret

Le tabouret a été réalisé l'aide d'un `BoxCollider2D` et d'un script lui permet de bloquer la progression d'une boîte mais pas celle du joueur.

**Listing 3.8 – Script du tabouret**

```

1 //Fonction OnCollisionEnter2D() appelé à chaque collision
2 /**
3 * <summary>Appelé à chaque collision : permet que le joueur ignore les collisions
4     avec cet objet, mais pas les autres objets</summary>
5 *
6 * <param name="collision">objet qui est entré en collision avec l'objet qui porte
7     le script</param>
8 */
9 * <returns>Return nothing</returns>
10 */
11 private void OnCollisionEnter2D(Collision2D collision)
12 {
13     if (collision.gameObject.tag == "Player")
14     {
15         Physics2D.IgnoreCollision(collision.gameObject.GetComponent<Collider2D>(),
16             this.GetComponent<Collider2D>());
17     }
18 }

```

```

14 }
15 }
```

Ici le script va regarder à chaque collision avec le BoxCollider2D si l'objet qui est rentré en collision porte le *Tag*<sup>9</sup> : « Player », si c'est le cas il le laisse entre dans sa boîte de collision, autrement non.

### La Benne à boîte

La benne à boîte a subit un redesign car l'ancien n'était pas assez explicite. Au fond de la poubelle, il y a maintenant de la lave qui permet de comprendre plus facilement qu'une boîte pousser sur la benne sera détruite.



FIGURE 3.85 – La nouvelle benne

Elle a également été réalisé à l'aide d'un BoxCollider2D et d'un script.

#### Listing 3.9 – Script de la benne à boîte

```

1 // Si qqc rentre en contact avec le dumpster
2 /**
3 * <summary>Détruit l'objet qui entre en collision si c'est une boîte (appelé auto
4 *         par unity)</summary>
5 * <param name="other">objet entrer en collision avec</param>
6 *
7 * <returns>Return nothing</returns>
8 */
9 private void OnTriggerEnter2D(Collider2D other)
10 {
11     //si le qqc est une boîte
12     if (other.tag == "Box")
13     {
14         //On détruit l'objet boîte.
15         Destroy(other.gameObject);
16     }
17 }
```

Lorsque l'objet entre en collision avec la benne, si c'est une boîte, il est détruit.

### 3.8.2 Crédit de la Salle

La scène utilisée est la même pour tous les niveaux du labyrinthe, les boîtes, tabourets et bennes sont posées par un script et le jeu possède une banque d'éénigmes stockée en fichier JSON<sup>10</sup>.

9. Étiquette en français

10. JSON est un format de données textuelles permettant de représenter de l'information structurée

## Design de la salle vide

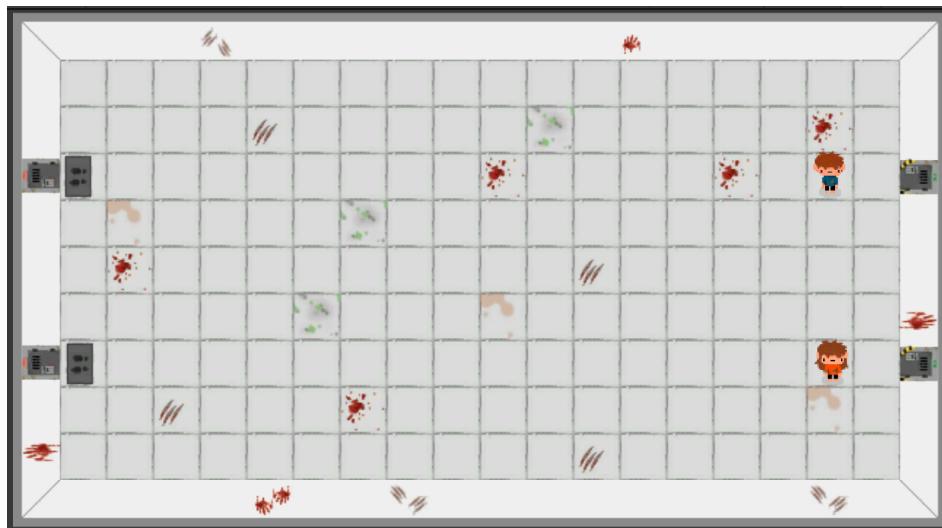


FIGURE 3.86 – La salle du labyrinthe de boîtes vide

La scène vide possède simplement le design (porte et sols), les murs qui empêche le joueur de sortir de la zone et les plaques de pression (codées lors de la première soutenance) qui permettent de se téléporter dans la salle suivante.

### Banque d'énigme

La première liste de coordonnées correspond aux boîtes qui bougent, la deuxième à celles qui ne bougent pas, la troisième aux tabourets et la dernière aux bennes.

#### Listing 3.10 – Script de la benne à boîte

```

1 private List<Vector3[]> getData(string filename)
2 {
3     //Lecture du fichier JSON
4     var fileContent = Resources.Load<TextAsset>("CreateRooms/" + filename);
5     //Analyse du fichier
6     // Cree une liste de liste de vector3
7     List<Vector3[]> roomData = JsonConvert.DeserializeObject<List<Vector3[]>>(
8         fileContent.text);
9     return roomData;
}

```

Le script charge les données sous forme de tableau de tableau de Vecteur3<sup>11</sup>. Les fichiers de données sont stocké dans un dossier ressource qui est compilé avec le jeu lors du build.

11. Permet de définir un Vecteur dans un espace tridimensionnel dans Unity

```

{}> room_4_1.json > [ ] 1
1   [
2     [
3       {
4         "x": -7, "y": 0, "z": 0 },
5         "x": -6, "y": 1, "z": 0 },
6         "x": -6, "y": -1, "z": 0 },
7         "x": -8, "y": 3, "z": 0 },
8         "x": -8, "y": 2, "z": 0 },
9         "x": -8, "y": 0, "z": 0 },
10        "x": -8, "y": -2, "z": 0 },
11        "x": -8, "y": -3, "z": 0 },
12        "x": -9, "y": 3, "z": 0 },
13        "x": -9, "y": 2, "z": 0 },
14        "x": -9, "y": 0, "z": 0 },
15        "x": -9, "y": -2, "z": 0 },
16        "x": -9, "y": -3, "z": 0 },
17        "x": -10, "y": 3, "z": 0 },
18        "x": -10, "y": 2, "z": 0 },
19        "x": -10, "y": 1, "z": 0 },
20        "x": -10, "y": 0, "z": 0 },
21        "x": -10, "y": -1, "z": 0 },
22        "x": -10, "y": -2, "z": 0 },
23        "x": -10, "y": -3, "z": 0 },
24        "x": -11, "y": 3, "z": 0 },
25        "x": -11, "y": 2, "z": 0 },
26        "x": -11, "y": 1, "z": 0 },
27        "x": -11, "y": 0, "z": 0 },
28        "x": -11, "y": -1, "z": 0 },
29        "x": -11, "y": -2, "z": 0 },
30        "x": -11, "y": -3, "z": 0 }
31    ],
32    [
33      {
34        "x": -5, "y": -4, "z": 0 },
35        "x": -5, "y": -3, "z": 0 },
36        "x": -5, "y": -2, "z": 0 },
37        "x": -5, "y": -1, "z": 0 },
38        "x": -5, "y": 1, "z": 0 },
39        "x": -5, "y": 2, "z": 0 },
40        "x": -5, "y": 3, "z": 0 },
41        "x": -5, "y": 4, "z": 0 },
42        "x": -9, "y": 4, "z": 0 },
43        "x": -9, "y": -4, "z": 0 },
44        "x": -12, "y": 1, "z": 0 },
45        "x": -12, "y": -1, "z": 0 },
46        "x": -7, "y": 1, "z": 0 },
47        "x": -7, "y": -1, "z": 0 },
48        "x": -8, "y": 1, "z": 0 },
49        "x": -8, "y": -1, "z": 0 },
50        "x": -9, "y": 1, "z": 0 },
51        "x": -9, "y": -1, "z": 0 }
52    ],
53    [
54      {
55        "x": -5, "y": 0, "z": 0 },
56        "x": -6, "y": 4, "z": 0 },
57        "x": -7, "y": 4, "z": 0 },
58        "x": -8, "y": 4, "z": 0 },
59        "x": -6, "y": -4, "z": 0 },
60        "x": -7, "y": -4, "z": 0 },
61        "x": -8, "y": -4, "z": 0 }
62    ],
63    [
64      {
65        "x": -11, "y": 4, "z": 0 },
66        "x": -10, "y": 4, "z": 0 },
67        "x": -11, "y": -4, "z": 0 },
68        "x": -10, "y": -4, "z": 0 }
69    ]
]

```

FIGURE 3.87 – Représentation d'une salle dans la base de données

Génération Automatique de la Salle

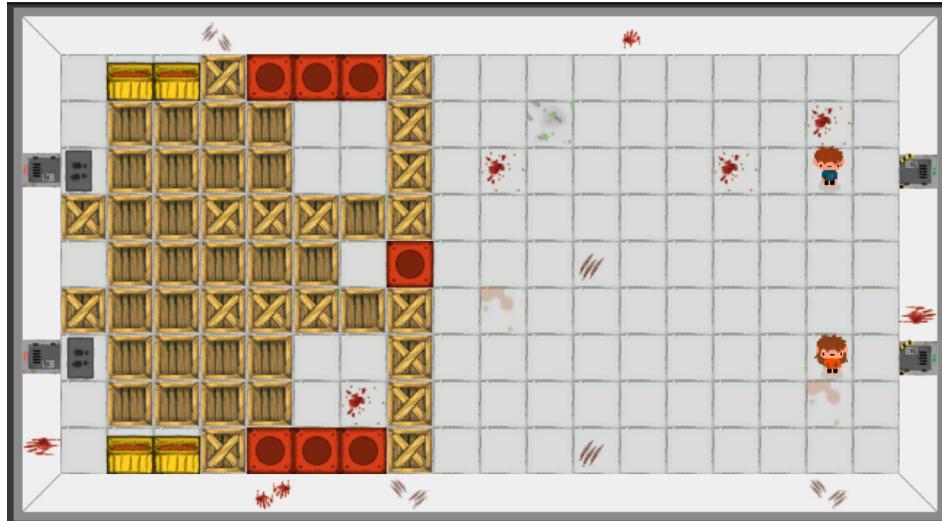


FIGURE 3.88 – La salle du labyrinthe de boîtes générée à partir de données de la figure ci-dessus

**Listing 3.11 – Script de la benne à boîte**

```
1 public void loadScene(string roomToLoad)
2 {
3     //Récupérer dans une variable le canvas d'interaction
4     GameObject canvaTextPopUP = GameObject.Find("TextPopUpCanvas");
5
6     //Chargement des coordonnées des boîtes de la pièce
7     List<Vector3 []> roomData = getData(roomToLoad);
8
9     //Instanciation des Boites qui bouge
10    foreach (Vector3 coord in roomData[0])
11    {
12        Instantiate(movableCratePrefab, new Vector3((float)0.32 * coord.x - (float)
13            0.16, (float)0.32 * coord.y + (float)0.16, 0), Quaternion.identity);
14    }
15
16    //Instanciation des Boites qui bouge pas
17    foreach (Vector3 coord in roomData[1])
18    {
19        Instantiate(unmovableCratePrefab, new Vector3((float)0.32 * coord.x - (
20            float)0.16, (float)0.32 * coord.y + (float)0.16, 0), Quaternion.
21            identity);
22    }
23
24    //Instanciation des tabourets
25    foreach (Vector3 coord in roomData[2])
26    {
27        Instantiate(stoolPrefab, new Vector3((float)0.32 * coord.x - (float)0.16,
28            float)0.32 * coord.y + (float)0.16, 0), Quaternion.identity);
29    }
30
31    //Instanciation des poubelles
32    foreach (Vector3 coord in roomData[3])
33    {
34        Instantiate(dumpsterPrefab, new Vector3((float)0.32 * coord.x - (float)
35            0.16, (float)0.32 * coord.y + (float)0.16, 0), Quaternion.identity);
36    }
37
38    //Lier aux boîtes qui bougent le canvas d'interaction
39    foreach (GameObject movableCrate in GameObject.FindGameObjectsWithTag("Box"))
40    {
41
```

```

36         movableCrate.GetComponent<MovableCrate>().MessageOnScreenCanvas =
            messageOnScreenCanvas;
37     }
38 }
```

Ce script va d'abord récupérer le lien vers le Canvas<sup>12</sup> tant qu'il n'y a pas beaucoup d'objets à analyser, puis récupérer les coordonnées et placer les objets à ces coordonnées. Enfin, il lie les boîte qui peuvent bouger au Canvas pour afficher un message au joueur lorsqu'il est près d'une boîte et souhaite la tirer.

### 3.9 Énigme : Le Labyrinthe Invisible

Cette partie a été réalisée par Timothy.

Le labyrinthe a été intégré dans Unity dans un niveau. Il a subit quelques modifications et a également été adapté pour le mode multijoueur en ligne. Cette énigme est entièrement fonctionnelle à l'exception de la perte de vie subit par les joueurs lorsqu'ils se trompent et marchent sur un mur. Le niveau dans lequel se trouve le labyrinthe n'est pas complètement finis : il reste les portes à intégrée ainsi que des boîtes et une benne de l'épreuve du labyrinthe des boîtes<sup>13</sup>.

#### 3.9.1 Modifications du labyrinthe

Depuis la précédente soutenance, plusieurs éléments constituant cette épreuve ont été améliorés. Le sprite<sup>14</sup> utilisé pour la carte du labyrinthe a été revu et modifié pour mieux coller avec l'ambiance générale et le thème du jeu.

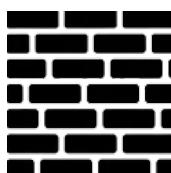


FIGURE 3.89 – Ancien Sprite



FIGURE 3.90 – Nouveau Sprite

Le système de détection utilisé pour vérifier que le joueur ne marche pas sur un mur a été revu. A l'origine j'utilisais les coordonnées du joueur pour calculer sa position dans le labyrinthe. Maintenant, des objets vides servant comme triggers et représentant les murs sont intanciés dans la scène. Cette méthode à plusieurs avantages :

- Elle demande moins de calcul à chaque frame<sup>15</sup> du jeu
- Elle est plus simple à intégrer dans un niveau du fait qu'elle utilise des objets représentant les murs dans la scène, contrairement à la méthode originale. De ce fait, la position, taille, orientation du labyrinthe est plus simple à manipuler.

#### 3.9.2 Intégration du multijoueur en ligne

Pour le mode en ligne, les scripts utilisés par le labyrinthe ont dû être modifiés. En effet, avec les scripts originaux, le labyrinthe se générat plusieurs fois<sup>16</sup>.

12. Element permettant d'afficher un objet à un endroit précis de l'écran du joueur

13. cf 3.8 Labyrinthe de Boîtes

14. Les sprites sont des images utilisés dans des jeux vidéos

15. Une frame est un instant dans le jeu

16. Une fois par joueur : la fonction void Start() s'exécutaient à chaque fois qu'un joueur rejoignait la scène

Pour régler cela, j'attend que les 2 joueurs rejoignent la scène, puis l'hôte de la partie<sup>17</sup> instancie dans la scène le labyrinthe et sa carte.

#### Listing 3.12 – Script du téléchargement de la dernière version du jeux

```

1 // Update is called every frame
2 void Update()
3 {
4     // Check if there are two players in the scene and if the player is the host,
        otherwise the labyrinthe is created twice
5     if (GameObject.FindGameObjectsWithTag("Player").Length == 2 && PhotonNetwork.
        IsMasterClient)
6     {
7         CreateLabyrinthe();
8     }
9 }
```

L'inconvénient de cette méthode est qu'elle ne fonctionne qu'avec 2 joueurs : en effet, une fois que les 2 premiers joueurs ont rejoint le niveau, les autres joueurs n'auront pas le labyrinthe et sa carte dans leur scène.

### 3.9.3 Améliorations à venir

Bien que l'énigme du labyrinthe soit finie, son niveau ne l'est pas. Il manque en effet l'intégration des porte pour changer de salle, ainsi que des boîtes, positionnées de manière bloquer la sortie et à forcer un des deux joueurs à franchir le labyrinthe.

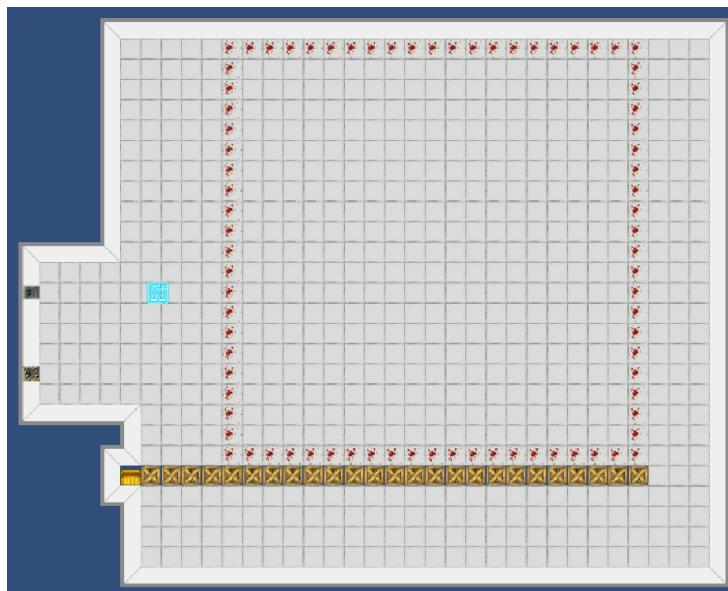


FIGURE 3.91 – Niveau du labyrinthe, le labyrinthe est délimité par les cases rouges, la map est accessible sur la case bleue

Dans le niveau ci-dessus, les joueurs arrivent à gauche et un joueur doit franchir le labyrinthe délimité par le carré rouge, afin des pousser les boîtes vers la gauche, libérant ainsi l'accès vers la sortie au second joueur.

---

17. La personne qui a créé la Room

### 3.10 Énigme : Labyrinthe fléché

Depuis la dernière soutenance la génération du labyrinthe a été finalisée et il a dorénavant toujours une solution grâce au script ci-dessous qui détermine un path fléché minimal.

Listing 3.13 – Script de la génération du labyrinthe

```

1 //setup entrance of laby
2 (int Xt, int Yt) = (0, Random.Range(0, y - 1));
3 laby[Xt, Yt] = 2010; //down but not up
4 Xt += 1;
5 int previous = (int)Direction.down;
6
7 //create logic laby
8 while(Xt < x)
9 {
10     int r = Random.Range(0, 3);
11     if (r == 0) // go down
12     {
13         laby[Xt, Yt] = (int)Direction.down+addPrevious(previous); //add needed
14         direction to tile
15         Xt += 1; //go down in the path
16         previous = (int)Direction.down;
17     }
18     else if (r == 1 && Yt + 1 < y && previous != (int)Direction.left) //go right
19     {
20         laby[Xt, Yt] = (int)Direction.right+addPrevious(previous);
21         Yt += 1;
22         previous = (int) Direction.right;
23     }
24     else if (r == 2 && Yt - 1 >= 0 && previous != (int)Direction.right)//go left
25     {
26         laby[Xt, Yt] = (int)Direction.left + addPrevious(previous);
27         Yt -= 1;
28         previous = (int) Direction.left;
29     }
}

```

Le résultat de ce script est un tableau qui possède les directions nécessaire pour la solution du labyrinthe.

Les tuiles correspondantes aux conditions de validités du labyrinthe sont ensuite affichées sur les deux joueurs sachant que chaque joueur ne voit pas le même ensemble de tuiles. En effet, pour favoriser la communication et l'entraide entre les joueurs, ils se repartissent les tuiles du labyrinthe. Le code s'occupant de cette partie est affiché ci-dessous.

Cela est fait en créant un nouveau tableau qui possédera un nombre entre 0 et 2, celui ci sera assigné aléatoirement et indiquera quel joueur verra la tuile. Nous parcourons ensuite toutes les tuiles du labyrinthe pour les afficher.

Listing 3.14 – Script de répartition des tuiles et affichage

```

1
2 /*
3 * Create a new array to determine which player show which tile
4 */
5 int[,] spawnLaby = new int[x, y];
6 int r = Random.Range(0, 3);
7
8 for (int x = 0; x < laby.GetLength(0); x++)
9 {
10     for (int y = 0; y < laby.GetLength(1); y++)
11     {
12         r = Random.Range(0, 8);
13         if(r == 1 || r == 0) //setup both
14         {
15             spawnLaby[x, y] = 2;
16         }
17         else if(r%2 == 0) //setup local
18         {

```

```

19         spawnLaby[x, y] = 0;
20     }
21     else //setup distant
22     {
23         spawnLaby[x, y] = 1;
24     }
25 }
26 }
27
28 //draw laby
29 int prefabNb;
30 GameObject t;
31 for (int x = 0; x < laby.GetLength(0); x++)
32 {
33     for (int y = 0; y < laby.GetLength(1); y++)
34     {
35         prefabNb = prefabRules(laby[x, y]);
36         laby[x, y] = toBin(prefabNb);
37         if(spawnLaby[x, y] == 0) //setup local
38         {
39             Instantiate(prefabsL[prefabNb], new Vector3(startX + y * 0.32f + 0.1f,
40                     startY - x * 0.32f - 0.32f), Quaternion.identity, parentObj.
41                     transform);
42             t = PhotonNetwork.Instantiate(prefabsL[0].name, new Vector3(startX + y
43                     * 0.32f + 0.1f, startY - x * 0.32f - 0.32f), Quaternion.identity);
44             t.SetActive(false);
45         }
46         else if(spawnLaby[x, y] == 1) //setup distant
47         {
48             Instantiate(prefabsL[0], new Vector3(startX + y * 0.32f + 0.1f, startY
49                     - x * 0.32f - 0.32f), Quaternion.identity, parentObj.transform);
50             t = PhotonNetwork.Instantiate(prefabsL[prefabNb].name, new Vector3(
51                     startX + y * 0.32f + 0.1f, startY - x * 0.32f - 0.32f), Quaternion.
52                     identity);
53             t.SetActive(false);
54         }
55     }
56 }
57 }
58 }
```

Les erreurs des joueurs sont aussi récupérées et traitées (les deux joueurs sont téléportés au début à chaque erreur commise par l'un des deux joueurs).

Ceci est permis grâce au script ci-dessous qui va comparer la position actuel et précédente du joueur pour déterminer une direction prise. Cette direction sera comparée aux tuiles d'avant et d'après pour voir si le mouvement est possible.

**Listing 3.15 – Script de la vérification des déplacements**

```

1 /*
2 * RespectRules vérifie qu'une tuile donnée permet le déplacement dans une direction
3 * donnée.
4 */
5 private bool CheckPlayerMovement(int[] playerTmpTile, int[] playerTile)
6 {
7     bool goToOut = false;
8     bool fromOut = false;
9
10    //check if exit or enter in laby
11    if (playerTile[1] < 0 || playerTile[0] < 0 || playerTile[1] >= y ||
12        playerTile[0] >= x) fromOut = true;
13    if (playerTmpTile[1] < 0 || playerTmpTile[0] < 0 || playerTmpTile[1] >= y
14        || playerTmpTile[0] >= x) goToOut = true;
15
16    int tile = 0;
17    int newTile = 0;
```

```

15         if (goToOut && fromOut) return true;
16
17         if (!fromOut) tile = laby[playerTile[1], playerTile[0]];
18
19         if (!goToOut) newTile = laby[playerTmpTile[1], playerTmpTile[0]];
20
21         if (playerTile[0] != playerTmpTile[0]) //change x
22     {
23             if (playerTile[0] > playerTmpTile[0]) //go left
24             {
25                 if (!fromOut && !RespectRules(tile, Direction.left)) return false;
26                     //previous tile has not left arrow
27                 if (!goToOut && RespectRules(newTile, Direction.right)) return
28                     false; //current tile has a right arrow
29             }
30             else //go right
31             {
32                 if (!fromOut && !RespectRules(tile, Direction.right)) return false;
33                 if (!goToOut && RespectRules(newTile, Direction.left)) return false
34                     ;
35             }
36         else if(playerTile[1] != playerTmpTile[1]) //change y
37     {
38             if (playerTile[1] > playerTmpTile[1]) //go up
39             {
40                 if (!fromOut && !RespectRules(tile, Direction.up)) return false;
41                     if (!goToOut && RespectRules(newTile, Direction.down)) return false
42                     ;
43             }
44             else //go down
45             {
46                 if (!fromOut && !RespectRules(tile, Direction.down)) return false;
47                 if (!goToOut && RespectRules(newTile, Direction.up)) return false;
48             }
49         }
50     }
51     return true;
52 }
```

---

Le multijoueur a été implémenté sur cette énigme et est totalement fonctionnel, les joueurs peuvent se voir et ils ont chacun une partie du labyrinthe différente comme montré sur les deux figures issue de la même session.

Les joueurs peuvent ensuite aller à la scène suivante en allant sur les plaques de pressions (visible en bas de la figure de gauche).

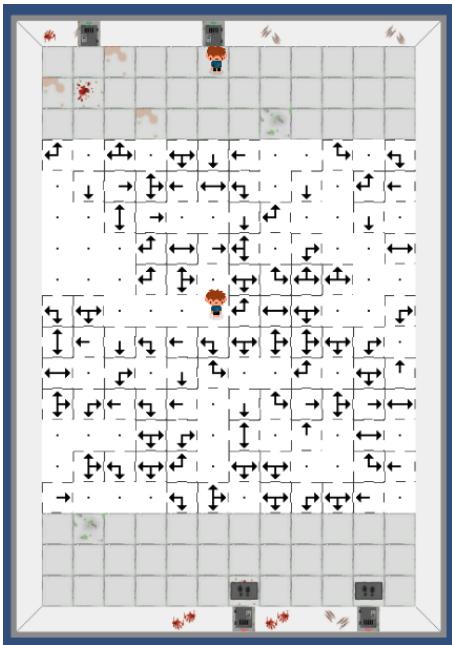


FIGURE 3.92 – Labyrinth master side

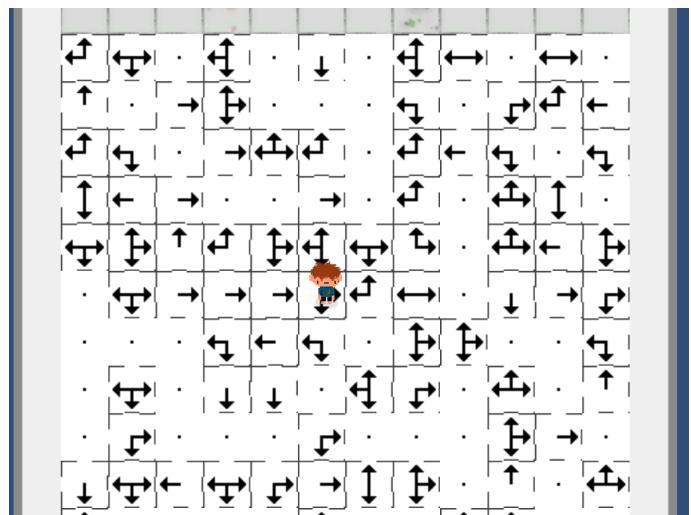


FIGURE 3.93 – Labyrinth client side

## 3.11 Narration

### 3.11.1 Rédaction du scénario

La rédaction du scénario a été réalisée par Ethan.

Une majorité des scripts ont déjà été écrit, notamment celui de l'introduction et de la conclusion, tous ceux de l'énigme du labyrinthe de boîtes et ceux d'avant et après la première rencontre avec un monstre.

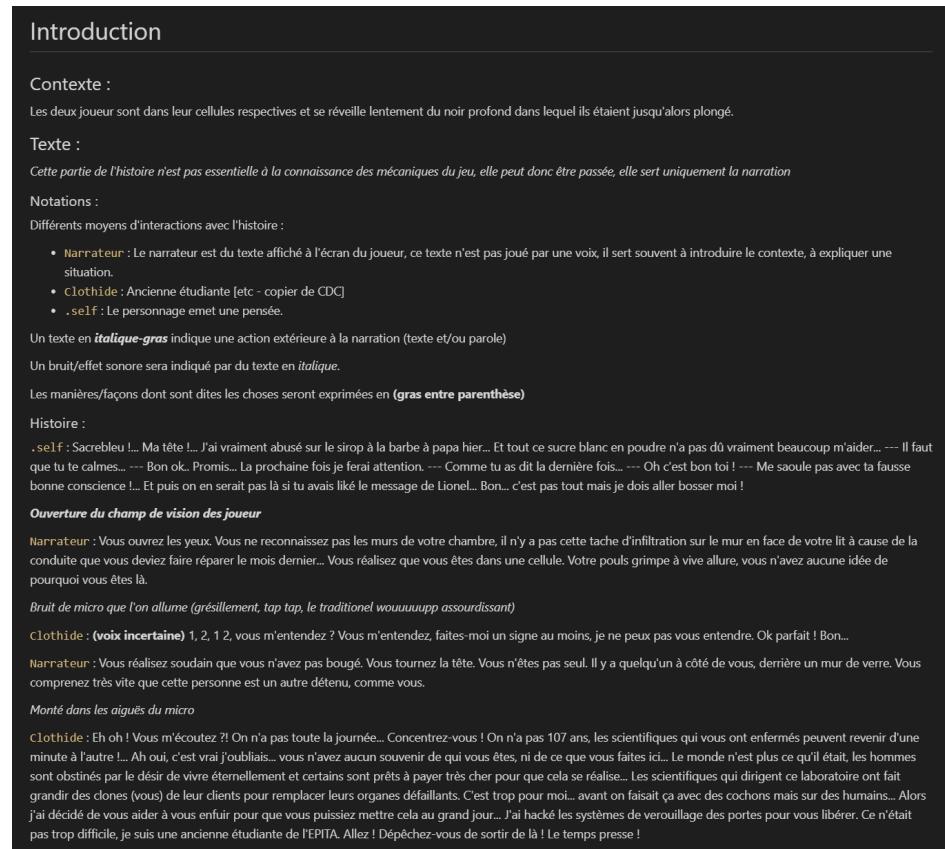


FIGURE 3.94 – Script de l'introduction

### 3.11.2 Système de Dialogue

Le système de dialogue est utile à la narration du jeu. C'est grâce à cela que les joueurs et les PNJ<sup>18</sup> du jeu s'interagissent ensemble. Il a été créé par Timothy et modifié par Ethan pour convenir aux besoins du jeu

### Création

Cette partie a été réalisée par Timothy.

Les dialogues utilisent un trigger placé dans divers niveaux pour se déclencher ainsi qu'une classe Dialogue afin d'enregistrer le nom de la personne qui parle ainsi que ce qu'elle dit. Le trigger a également une variable de type Dialogue afin de pouvoir déclencher un dialogue différent pour chaque trigger.

#### Listing 3.16 – Classe Dialogue

```

1 public class Dialogue
2 {
3     // Name of the speaker
4     public string name;
5     // Sentences that the character say
6     public string[] sentences;
7 }
```

Chaque phrase présente dans le dialogue est enregistrée dans une File<sup>19</sup>, permettant ainsi de facilement accéder à la phrase à afficher dans le canvas, avec une animation où chaque lettre apparaît un

18. Personnage Non Joueur

19. Une File type abstrait basée sur le principe « premier entré, premier sorti »

court instant après la précédente. Cet instant de « pause » est géré par une Coroutine<sup>20</sup> en utilisant du mot clé yield<sup>21</sup>.

**Listing 3.17 – Affichage d'une phrase**

```

1 // Beginning of the Coroutine & call of function TypeSentence()
2 StartCoroutine(TypeSentence(sentence));
3
4 IEnumerator TypeSentence (string sentence)
5 {
6     // Add text to canvas (dialogueText is the text field of the dialogue in the
7     // canvas)
8     dialogueText.text = "";
9
10    // sentences est la File contenant les phrases à afficher
11    foreach (char letter in sentences.Dequeue())
12    {
13        // Append letter to canvas
14        dialogueText.text += letter;
15        // Restart Coroutine to make a break
16        yield return null;
17    }

```

### Importation du texte depuis des fichiers JSON

Cette partie a été réalisée par Ethan.

J'ai modifié le système de dialogue de Timothy pour que le système dialogue aille chercher le phrases du script dans un fichier JSON et qu'il supporte le fait qu'il y ait potentiellement plusieurs personnes participant à la discussion.

Ci-dessous un exemple de fichier JSON stockant les informations nécessaires au système de dialogue, à savoir : le nom du locuteur, son discours séparé en un tableau de phrases et le lien (éventuel) vers le discours suivant (d'une autre personne).

---

20. Une Coroutine est une méthode permettant de pauser l'exécution d'un code et de le reprendre plus tard  
 21. Yield permet de retourner plusieurs éléments en un appel

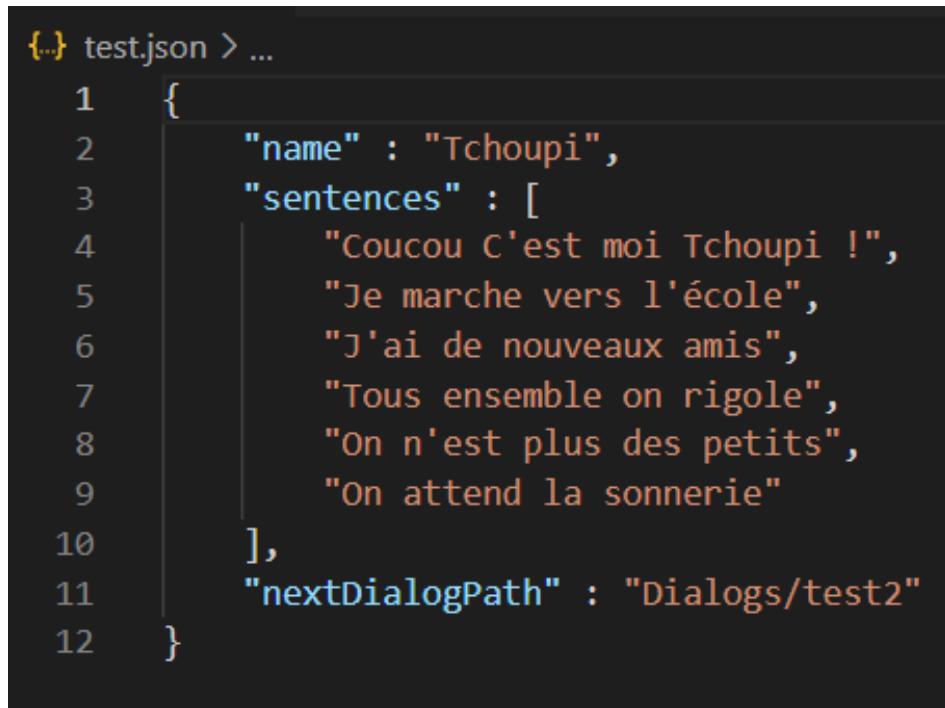


FIGURE 3.95 – Exemple d'un dialogue de Tchoupi

Listing 3.18 – Cr éation du labyrinthe

```

1 public void TriggerDialogue()
2 {
3   if (filePath != "") {
4     //Lecture du fichier json
5     var dialoguesData = Resources.Load<TextAsset>(filePath);
6
7     //cr éation de l'objet qui stocke les phrases du dialogue en fonction du json
8     Dialogue dialogue = JsonUtility.FromJson<Dialogue>(dialoguesData.text);
9
10    //changement du chemin du nom du prochain fichier
11    filePath = dialogue.nextDialogPath;
12
13    //D茅but du dialogue
14    FindObjectOfType<DialogueManager>().StartDialogue(dialogue, this);
15  }
16  else if (objectsToActivate != null && objectsToActivate.Count != 0)
17  {
18    foreach (GameObject obj in objectsToActivate)
19    {
20      obj.SetActive(true);
21    }
22  }
23 }

```

Le script permet de charger le fichier et de r éarmer le script avec l'eventuel suivant dans le cas ou la string du suivant est vide, alors le dialogue est finit et on active les objet qui permettent de passer à la salle suivante.

## 3.12 Site Web (Front)

Cette partie a été réalisé dans son intégralité par Ethan.

J'ai d'abord réalisé tout un plan du site que l'on a validé ensemble avec toute l'équipe.

Les efforts ont principalement été centré sur le design.

Ont pour l'instant été réalisé :

- La NavBar (pour naviguer d'une page à l'autre).
- Une Pop-Up intempestive qui propose de télécharger notre magnifique jeu .
- La page de présentation des membres de l'équipe.
- La page de présentation du scénario du jeu.

Il ne reste plus que la page d'accueil pour laquelle il était nécessaire d'avoir des images du jeu fini et pour les autres pages, ce n'est que du contenu à créer, le style étant déjà fait.

Le site est réalisé en pur HTML<sup>22</sup>/CSS<sup>23</sup> et JavaScript<sup>24</sup>.

Ci-dessous, quelques images du site.

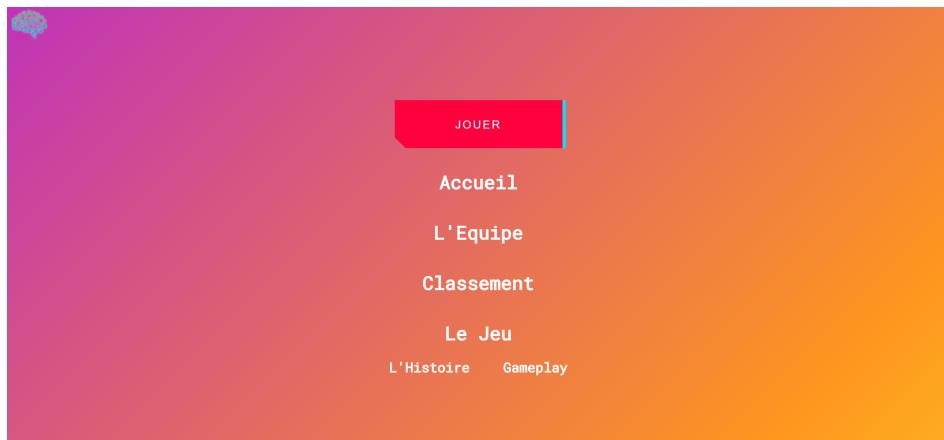


FIGURE 3.96 – Navigation du site



FIGURE 3.97 – Extrait de la page de présentation du scénario



FIGURE 3.98 – Titre de la page de présentation du groupe

22. HTML est un langage de balisage conçu pour représenter les pages web

23. CSS est un langage informatique qui décrit la présentation des documents HTML et XML

24. JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives

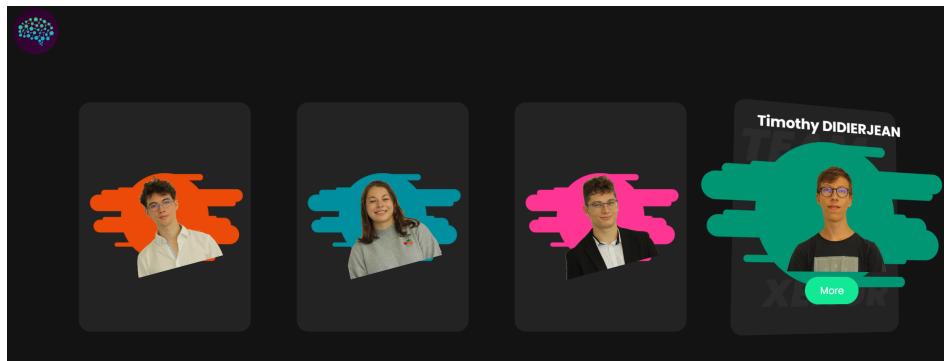


FIGURE 3.99 – Carte des membres qui bougent avec la souris



FIGURE 3.100 – Texte déployable avec un bouton sur la carte

### 3.13 Site Web (Back)

Cette partie a été réalisé par Hugo.

Le backend du site a pour mission de récupérer les scores des joueurs à partir d'Unity et de les envoyer au frontend du site lorsqu'il est chargé.

Pour stocker les scores nous utilisons une base de donnée MySQL<sup>25</sup>.

---

25. MySQL est un système de gestion de bases de données relationnelles

## Design de la salle vide

		<b>id</b>	<b>user1</b>	<b>user2</b>	<b>score</b>
	Edit	1	&quot;gerard&quot;	&quot;gerture&quot;	500
	Edit	2	gerard	gerture	500
	Edit	3	gerard	gerture	500
	Edit	4	gerard	gerture	500
	Edit	5	gerard	gerture	500
	Edit	6	gerard	gerture	500
	Edit	7	gerard	gerture	500
	Edit	8	gerard	gerture	500
	Edit	9	gerard	gerture	500
	Edit	10	gerard	gerture	500
	Edit	11	gerard	gerture	500
	Edit	12	gerard	gerture	500
	Edit	13	gerard	gerture	500

FIGURE 3.101 – Vu de la base de donnée depuis PHPMyAdmin<sup>26</sup> rempli avec des données temporaires

Le premier script PHP<sup>27</sup> permet de charger les données à partir d'une requête GET qui pourra ou non contenir un offset afin d'obtenir les données à partir d'un certain rang.

Listing 3.19 – Script load.php

```

1 <?php
2 include('config.php');
3
4 //allow CORS
5 header('Access-Control-Allow-Origin: *');
6 header('Access-Control-Allow-Methods: GET, POST');
7 header("Access-Control-Allow-Headers: X-Requested-With");
8
9
10 $offset = 0;
11
12 //get offset is exist and not empty
13 if(isset($_GET["offset"]) && !empty($_GET["offset"])){
14     $offset = htmlspecialchars($_GET["offset"]);
15 }
16
17
18 //requests to bdd to fetch 10 element based starting at the offset
19 // $req = $bdd->prepare("SELECT * FROM scores ORDER BY score ASC LIMIT 10 OFFSET ".
20 //     "$offset.");
21 $req = $bdd->prepare("SELECT * FROM scores LIMIT 10 OFFSET ".$offset."");
22 $req->execute(array());
23
24 $info = $req->fetchAll();
25
26 header('Content-Type: application/json');
27 echo json_encode($info);
28 ?>

```

Le second script PHP permet d'ajouter des scores à la base de donnée depuis une requête GET contenant les pseudos des deux joueurs et leur score.

Listing 3.20 – Script add.php

```

1 <?php
2 include('config.php');
3
4 //check if values are present and not empty

```

27. PHP est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP

```

5 if(isset($_GET["u1"]) && isset($_GET["u2"]) && isset($_GET["score"]) && !empty(
6     $_GET["u1"]) && !empty($_GET["u2"]) && !empty($_GET["score"])){
7     $u1 = htmlspecialchars($_GET["u1"]);
8     $u2 = htmlspecialchars($_GET["u2"]);
9     $score = htmlspecialchars($_GET["score"]);
10
11    //insert in bdd
12    $addScore = $bdd->prepare("INSERT INTO scores (user1, user2, score) VALUES (?, ?, ?)");
13    $addScore->execute(array($u1, $u2, $score));
14    header("HTTP/1.1 200 OK");
15 }else{
16     //respond with error
17     header("HTTP/1.1 400 Bad Request");
18 }
19 ?>

```

Les codes sont totalement fonctionnel en local mais ils ne sont pas encore publiés sur un serveur. Le page HTML utilisant les données est aussi fonctionnel mais le design n'est pas encore fait.

### 3.14 Launcher

Cette partie a été réalisé par Hugo.

Le launcher a trois mission :

- Installer le jeux
- Faire une mise à jour du jeux
- Désinstaller le jeux

Le launcher a été commencé en C#, il s'exécute sous la forme d'une console (visible ci dessous). Pour l'instant, il fait l'installation du jeux depuis les releases Github dans un dossier fournis (ou non) par l'utilisateur puis il ajoute un raccourci du jeu sur le bureau.

Il manque encore l'étape de l'ajout du jeux dans le Registre de Windows<sup>28</sup> (afin de le retrouver dans le panneau de contrôle) et la mémorisation du chemin d'installation (pour pouvoir le désinstaller). Une fois que cela sera fait, la mise à jour et la désinstallation se feront sans aucun soucis pour la dernière soutenance.

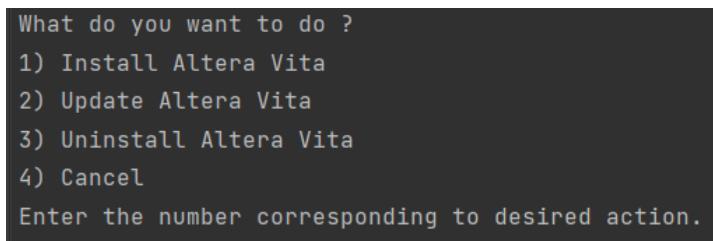


FIGURE 3.102 – Console du launcher

Le système de release Github permet d'héberger les binaires du jeux et grâce à l'API<sup>29</sup> de la plateforme et de la librairie C# Octokit<sup>30</sup> il est simple de télécharger la dernière version du jeu comme montré ci-dessous.

**Listing 3.21 – Cr éation du labyrinthe**

```

1 GitHubClient client = new GitHubClient(new ProductHeaderValue("SomeName"));

```

28. Le Registre Windows (ou Regedit) est une base de données définie par le système dans laquelle les applications et les composants système stockent et récupèrent les données de configuration

29. API (ou interface de programmation d'applications) est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels

30. Octokit est le client officiel de l'API Github

```

2 IReadOnlyList<Release> releases = await client.Repository.Release.GetAll("s2xenor",
    "xenor");
3 var latest = releases[0];
4
5 WebClient webClient = new WebClient();
6 webClient.Headers.Add("user-agent", "Anything");
7 await webClient.DownloadFileTaskAsync(new Uri(latest.Assets[0].BrowserDownloadUrl),
    destination + "/tmp.zip");
8 ZipFile.ExtractToDirectory(destination + "/tmp.zip", destination + "/AlteraVita");

```

### 3.15 Résumé de l'Avancement

TABLE 3.1 – Résumé de l'Avancement

Tâches	État
Création Menu Principal	F
Création du Lobby	F
Création des Cellules	C
Ajout du multijoueur en ligne	F
Intégration de l'énigme : « Enigme des fils »	C
Intégration de l'énigme : « Labyrinthe de Boites »	C : Manque le Mode Multijoueurs
Intégration de l'énigme : « Labyrinthe Fléché »	F
Intégration de l'énigme : « Labyrinthe Invisible »	F
Intégration de l'énigme : « Connecte les Produits Chimiques »	N
Création du menu principal	F
Intégration du système de dialogue	F
Intégration des bruitages	C
Création du Front-End du site	C
Création du Back-End du site	C : local seulement C : installation partiel
Création du launcher	C : installation partiel
Création des monstres	C : finalisation avec le système de combat
Création des potions	C : finalisation avec le système de combat
Système de combat	N
Implémentation des animations et des mouvements des joueurs mâle et femelle	F
Création de la barre de vie des joueurs	C : finalisation avec le système de combat

Légende :

Symbole	Signification
Fini	F
En Cours	C
Pas Encore Réalisé	N

### 3.16 Conclusion

Légende :

Toutes ces tâches ont été réalisé avec succès par rapport aux attentes.

TABLE 3.2 – Répartition des Responsabilités

Tâches	Elya	Ethan	Hugo	Timothy
Création du Lobby		A		
Intégration des premières énigmes dans le jeu			A	A
Création des premiers monstres.	A			A
Création des premières potions.	A			A
Création du système de Narration.		A		
Implémentation de l'énigme du labyrinthe à boîtes.		A		
Début de création du texte de la narration.		A		
Début de création du site (FrontEnd)		A		
Début de création du site (BackEnd)			A	
Création des assets principaux (sols, murs, portes)	A			
Début création du launcher			A	
Début création système multijoueur				A
Début recherche musique				A

Symbole	Signification
Atteint	A

# Chapitre 4

## Prévision

Pour la soutenance finale, nous devrons avoir réalisé ces différents points :

TABLE 4.1 – Répartition des Responsabilités pour la Soutenance Finale

Tâches	Elya	Ethan	Hugo	Timothy
Ajout des musiques				R
Création et Implémentation de l'énigme « Connecte les produits chimiques »		R		
Intégration des dernières énigmes dans le jeu			R	
Mise en place du système de combat joueur contre monstres	R			
Asset Spécifiques (détails, objet)	R			
Finalisation du FronEnd du site		R		
Finalisation du backend du site			R	
Fin d'implémentation système multijoueur				R
Finalisation du launcher			R	
Finalisation et implémentation du script de narration		R		

Légende :

Symbol	Signification
Responsable	R
Suppléant	S

# Chapitre 5

## Ressources

Outils et Plateformes utilisés :

**Unity** • Unity est un moteur de jeu multiplateforme. Il est très populaire dans l'industrie du jeu vidéo, aussi bien pour les grands studios que pour les indépendants.  
Nous utiliserons sa version gratuite.

**Github** • GitHub est un service d'hébergement de code et de gestion de développement de logiciels. Cela nous permettra de pouvoir accéder au fichier du projet facilement, et permettra de coder en simultané sans risque de perte. Cela simplifiera aussi la communication grâce aux issues, pull request, et project view disponible avec GitHub.

**Plateforme du CRI** • Le projet sera aussi synchronisé sur la plateforme du CRI.

**Photoshop** • Photoshop est un logiciel de retouche photo, de traitement d'image et de dessin assisté par ordinateur.  
Ce logiciel nous sert à créer le logo du jeu ainsi que certains assets.

**Bosca Ceoil** • Bosca Ceoil est un logiciel gratuit et simple à prendre en main de création de musique.  
Il nous servira pour réaliser les bruitage de notre jeu.

**Overleaf** • Overleaf est un éditeur LaTeX en ligne, collaboratif en temps réel.  
Il nous servira pour rédiger les diverses comptes-rendus demandés.

**Discord** • Discord est un logiciel gratuit de messagerie instantanée.  
Il nous sert à communiquer entre nous, de nous tenir au courant de l'avancée du projet, de poser des questions si un membre s'en pose.

# Chapitre 6

## Conclusion

Ainsi nous sommes une fois encore très satisfait de notre avancement sur le projet. En effet, nous avons réussi à réaliser l'intégralité de nos objectifs. Le jeu commence à prendre forme et nous sommes enjoués de son rendu. Cependant pour réalisés nos objectifs nous avons dû travailler régulièrement entre les deux soutenances et de façon plus soutenue qu'une majorité des autres groupes, nous nous rendons ainsi compte que nous avons été assez ambitieux dans nos objectifs et que par conséquent nous devons beaucoup travailler pour réaliser ces objectifs. Mais « Ce qui ne me détruit pas me rend plus fort. » comme le dis si bien Friedrich Nietzsche, alors nous ressortirons plus fort de cette expérience où nous aurons pu goûter à cette expérience tant recherchée par tous les développeurs de jeu-vidéos, nous parlons bien sûr ici du « Crunch<sup>1</sup> ». Nous sommes à la dernière ligne droite avant la fin du projet, à nous de ne rien lâcher.

---

1. Crunch est le terme communément utilisé par l'industrie du logiciel et l'industrie vidéoludique pour désigner une période intense de travail, généralement avant le rendu d'un jalon de projet