

Rapport de soutenance: Final



Léo CAPMARTIN, Thomas CORBIERE
Hugo SAINT GERMES, Olivier TALANE

Juin 2021

Table des matières

1	Introduction	4
2	Répartition des tâches	5
2.1	Répartition globale du travail	5
2.2	Répartition des énigmes	5
3	Description du jeu	6
3.1	Fonctionnement du jeu	6
3.1.1	Les règles	6
3.1.2	Le déroulement d'une partie	7
3.1.3	Composantes du jeu	7
3.2	Description des tâches	8
3.2.1	Les Énigmes	8
3.2.2	Les Missions	9
4	Multijoueur	10
4.1	La mise en place d'un serveur	10
4.2	Installation d'un premier "jeu" multijoueur	10
4.3	Finalisation du multijoueur	11
4.3.1	Création des rooms	11
4.3.2	Le lobby	12
4.3.3	En jeu	13
5	Physique du jeu	14
5.1	Les déplacements	14
5.1.1	Mouvements de base	14
5.1.2	Interactions de base avec un objet	15

6 Game Design	17
6.1 Interface utilisateur	17
6.2 La modélisation 3D	19
7 Algorithme de résolution	20
7.1 Principe	20
7.2 Fonctionnement	21
7.3 Implémentation	21
8 Mise en places des énigmes et missions	23
8.1 L'alchimie	23
8.2 L'échiquier	24
8.3 Enigme visuelle et calcul mental	25
8.4 Bibliothèque	27
8.5 Les câbles	28
8.6 Les horloges	29
8.7 Enigme sonore	31
8.8 Éléments à ouvrir	32
8.9 Les peintures	33
8.10 Le puzzle	34
9 Site Web	36
9.1 Outil utilisé	36
9.2 Mise en page	36
9.3 Hébergement	39
10 Trailer	40
11 Difficultés rencontrées	41
11.1 Site web	41
11.2 Physique du jeu	41
11.3 Enigmes	42
12 Ressenti personnel sur le projet	44
12.1 Thomas	44
12.2 Olivier	44
12.3 Hugo	45
12.4 Léo	45

13 Logiciels utilisés **47**

14 Conclusion **49**

Introduction

Bienvenue au coeur du projet Synapse, un jeu basé sur la réflexion et la compétition, le tout dans une ambiance médiévale.

Nous sommes le groupe Endless Code Studio composé de Léo Capmartin, Hugo Saint Germes, Olivier Talane et Thomas Corbiere, et nous sommes heureux de vous présenter la version finale de notre jeu.

Grâce à la bonne dynamique que nous avons dans le groupe, nous avons réussi à faire de Synapse une représentation fidèle de ce que nous avions imaginé lors de sa conception. Les énigmes composant le jeu sont variées, aussi bien sonores que visuelles. Le jeu étant basé sur la compétition, il est composé d'un multijoueur avec deux joueurs qui s'affrontent en temps réel dans des salles indépendantes où seul le premier joueur à avoir fini ses énigmes remporte la partie. De ce fait, les joueurs auront de quoi s'amuser longtemps sur notre jeu.

Tout au long de ce rapport, vous trouverez toutes les informations détaillées allant de la conception de notre jeu jusqu'aux ressentis personnels de chaque personne du groupe.

Bon voyage

Répartition des tâches

2.1 Répartition globale du travail

	Multijoueur	Énigme	Physique	Menu/Interface	IA	site
Léo	X	X			X	
Olivier		X		X		
Thomas		X	X			X
Hugo	X	X				

2.2 Répartition des énigmes

- **Thomas** : alchimie, bibliothèque, calcul mental, énigme visuelle, cable
- **Olivier** : énigme sonore
- **Hugo** : échiquier, horloge, éléments à ouvrir avec coffre
- **Léo** : puzzle, tableau

Description du jeu

3.1 Fonctionnement du jeu

Synapse est un jeu qui met en compétition deux joueurs. Chacun apparaît dans une salle d'éénigmes générées aléatoirement. La salle est la même pour les deux joueurs mais ils n'ont aucun moyen de se voir. Le premier qui complète ses éénigmes gagne la partie !

3.1.1 Les règles

Nous allons dans cette sous-partie vous décrire le fonctionnement en détail ainsi que les règles qui régissent notre jeu, à commencer par le fonctionnement des salles.

Une escape game classique est composée d'[éénigmes](#) et de [missions](#). Les éénigmes sont de petits puzzles plus ou moins complexes à résoudre. En général, elles offrent une récompense au joueur lui permettant d'avancer sur la résolution des missions. Les missions sont des composantes faciles à résoudre. Lorsque le joueur les voit, il sait exactement quels seront les éléments de la solution (par exemple une clé ou une pièce de puzzle).

Dans notre jeu, une salle est donc composée de différentes éénigmes et missions. Ces différentes composantes forment un arbre que l'on remontera afin de résoudre l'élément présent à la racine. Le joueur résoudra d'abord les éénigmes et les jeux, puis grâce aux composants fournis par ceux-ci, il pourra résoudre les missions et enfin résoudre la salle.

3.1.2 Le déroulement d'une partie

Une partie nécessite deux joueurs. Pour jouer ensemble, l'un d'entre eux devra créer une partie que l'autre pourra rejoindre au moyen d'un code fourni au créateur. Une fois connectés ensemble, la partie se lance et un algorithme de résolution va générer une salle selon les règles établies ci-dessus en pi-ochant aléatoirement dans une collection prédefinie d'énigmes et missions. Lorsque la salle est générée, elle est dupliquée et chacun des joueurs est placé dans une des deux salles. Ils vont donc commencer à tenter de résoudre leur salle, le premier à y arriver est vainqueur de la partie.

3.1.3 Composantes du jeu

Premièrement le jeu est composé de deux salles. Ces deux salles seront par la suite remplies d'énigmes et de missions par notre algorithme de génération de salle. A la fin de l'exécution de l'algorithme, les 2 deux salles seront parfaitement identiques (cf [le déroulement d'une partie](#) pour plus de détails).

Dans nos salles, plusieurs éléments seront interactifs. Nous aurons par exemple des objets que l'on peut attraper (comme les potions de l'énigme "alchimie"). D'autres éléments déclencheront des évènements beaucoup plus complexes (comme une partie d'échecs).

Tous les éléments avec lesquels il est possible d'interagir sont précisés dans les parties "[Description des tâches](#)" et "[Mise en place des énigmes et missions](#)".

Enfin notre jeu comporte deux joueurs. Ceux-ci ont exactement les mêmes caractéristiques. Ils peuvent tout d'abord se déplacer au moyen des touches directionnelles du clavier ou alors des touches "ZQSD" (Z : avancer, Q : aller à gauche, S : reculer, D : aller à droite).

Les joueurs peuvent aussi regarder autour d'eux en déplaçant leur caméra de haut en bas et de droite à gauche. Pour ce faire ils auront juste à déplacer la souris et la caméra suivra le mouvement.

De plus, la touche "escape" permet de faire apparaître un menu et à de faire quitter les énigmes. La touche "espace" permet au joueur de sauter lorsqu'il est sur le sol.

Enfin, grâce à la touche "E", les joueurs peuvent interagir avec leur environnement. Lorsqu'ils tiennent un objet en main, le clic gauche leur permettra de le lâcher devant eux.

3.2 Description des tâches

3.2.1 Les Énigmes

- **Tableaux à remettre à la bonne place :** Des tableaux seront placés dans la salle. Le joueur devra les remettre à la bonne place grâce à un indice sur chaque cadre.
- **Énigmes visuelles :** Des charades seront placées dans la pièce. Le joueur devra trouver la réponse du problème et l'écrire dans un carnet.
- **Énigmes sonores :** Grâce à un enregistrement audio, le joueur devra activer des leviers en fonction du son joué. Si le son est aigu, le levier devra être en position haute sinon le levier devra rester à sa position de base. Le joueur pourra ensuite valider l'énigme grâce à un bouton.
- **Déplacement d'éléments :** Des éléments du décor seront déplaçables, ce qui permettra de révéler des objets essentiels dans la résolution de certaines énigmes.
- **Remettre l'heure :** Une horloge sera placée dans la salle. Le joueur devra la remettre à la bonne heure (celle-ci sera donnée par un autre objet du décor).
- **Ranger une bibliothèque :** Le joueur aura des livres dans la bibliothèque et pourra les inverser pour les remettre en ordre.
- **Jeu d'échecs :** Un échiquier sera soumis au joueur et il devra déplacer deux pièces pour mettre en échec et mat le roi adverse. S'il se trompe le plateau sera réinitialisé.

- **Calcul mental** : Cinq calculs seront posés au joueur. Les réponses pour chacun des calculs seront demandées sous forme de QCM. S'il se trompe, tout sera remis à 0. L'ordre des calculs est déterminé de manière aléatoire.

3.2.2 Les Missions

Les éléments de solutions des missions seront donnés par les énigmes et les jeux.

- **Éléments à ouvrir** : Un coffre et une porte seront à ouvrir avec une clé. La clé sera cachée dans la salle.
- **Boîtier électrique** : Des fils électriques devront être branchés au bon endroit pour déclencher un évènement.
- **L'alchimie** : Il y aura des fioles avec des liquides à récupérer et à combiner.
- **Puzzle** : Grâce aux énigmes résolues le joueur pourra récupérer des pièces qui lui permettront de résoudre le puzzle.

Multijoueur

4.1 La mise en place d'un serveur

Pour le système multijoueur, nous avons fait le choix d'utiliser un asset d'Unity : Photon (PUN 2), nous avons décidé de prendre ce dernier car il met à disposition un serveur en ligne gratuitement. Il nous permet également d'avoir une facilité sur la gestion du lobby ou des rooms. Si on considère le jeu comme étant une ville, une room peut être vue comme un bâtiment dans lequel des joueurs peuvent jouer ensemble. Un lobby peut être vu comme la salle d'attente d'une room, tant que le nombre requis de joueurs connectés n'est pas atteint, on reste dans le lobby. Hugo s'est chargé de la mise en place de Photon et de l'association à un serveur sur le projet, cette partie là était assez simple. La partie la plus délicate aura été de comprendre qu'est-ce qu'un lobby et une room, comment ça fonctionne avec plusieurs personnes, et comment ça peut être implémenté.

4.2 Installation d'un premier "jeu" multijoueur

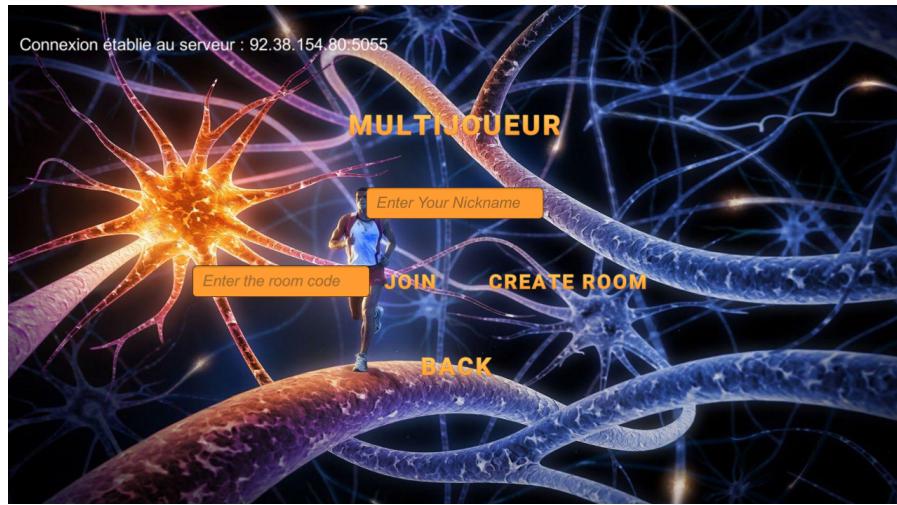
Après l'ajout de Photon et du serveur associé sur un projet Unity, Hugo s'est occupé de faire un "jeu" dans lequel on peut se déplacer et tourner avec une caméra commune pour tous les joueurs. Les deux joueurs n'étaient que de simples cylindres pouvant simplement se déplacer (sans saut ni caméra personnelle) sur une petite plateforme, cela a permis de se focaliser uniquement sur l'implémentation des premiers déplacements en ligne. Pour ce jeu, rien d'exceptionnel, quand on lance le jeu, on se connecte au serveur, puis dans un lobby, et si on appuie sur le bouton "Jouer", on va dans une

room prédefinie dans le code en faisant apparaître le joueur sur l'un des points d'apparition prédefini (l'implémentation finale des rooms a été faite par Léo). C'est avec ce que l'on a appris sur ce petit projet que l'on a construit notre jeu.

4.3 Finalisation du multijoueur

4.3.1 Création des rooms

Une fois le serveur mis en place avec Photon 2, nous devions créer un système de room plus avancé. Photon nous permet d'implémenter ce système assez facilement grâce aux fonctions fournies par la bibliothèque de Photon. Ainsi, nous avons pu implémenter un menu de connexion. Celui-ci est composé de trois parties : un champ permettant d'entrer un pseudonyme, une section permettant de créer une room et une section permettant de rejoindre une room déjà existante. Lorsqu'un joueur crée une room, il est automatiquement connecté à celle-ci. Pour rejoindre une room, un joueur doit renseigner le nom de celle-ci. Le nom d'une room est composée de six lettres et est généré aléatoirement lors de la création de la room.

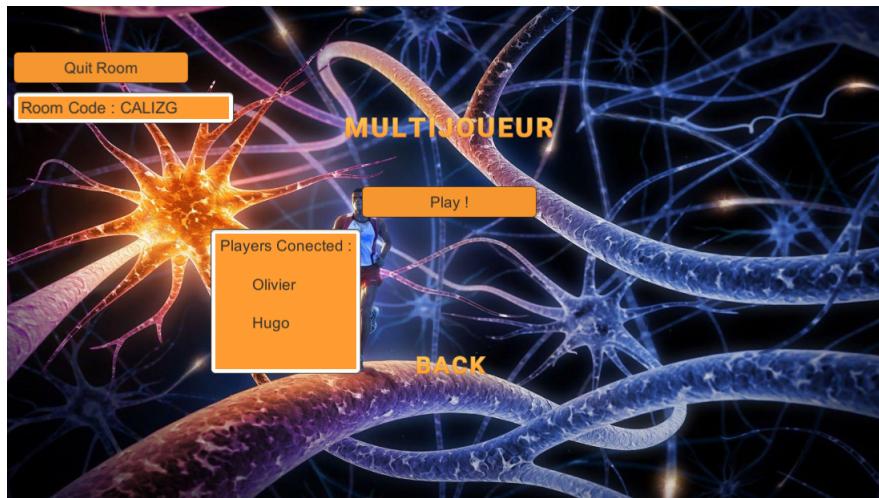


4.3.2 Le lobby

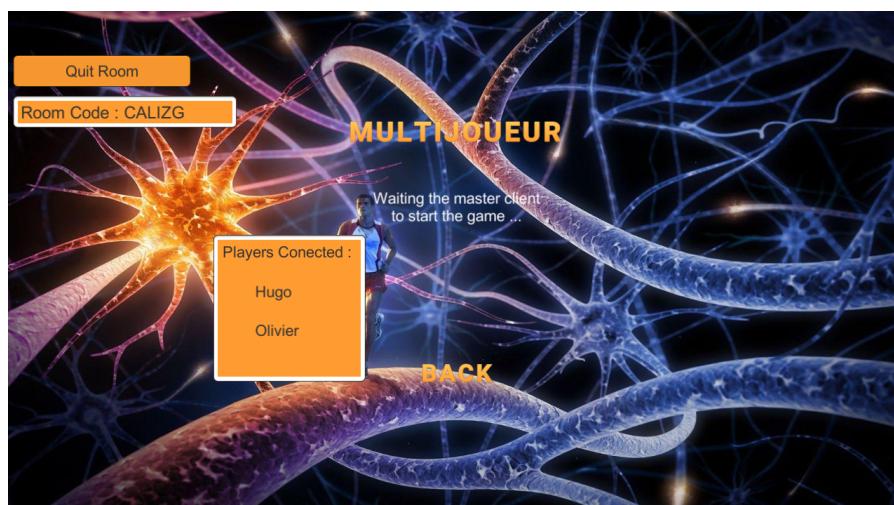
Une fois que le joueur entre dans une room, il est redirigé vers un autre menu, le lobby, sur lequel est affiché la liste des joueurs, le nom de la room, un bouton permettant de quitter la room et, s'il a lui-même créé la room, un bouton "Jouer". Ce bouton permet de lancer le jeu uniquement s'il y a le nombre de joueurs requis dans la room, en l'occurrence deux.

Lorsqu'un des deux joueurs quitte la room, si ce joueur est le créateur, l'autre joueur est automatiquement déconnecté, sinon, la liste des joueurs présents est actualisée.

Voici un exemple de ces deux menus :



Vue depuis le créateur de la room



Vue depuis le second joueur de la room

4.3.3 En jeu

Il fallait donc commencer par implémenter un vrai joueur avec de vrais contrôles et une caméra pour chaque joueur. Nous avons donc tenté d'implémenter le joueur réalisé par Thomas. Cependant, les deux joueurs ayant le même code, chacun pouvait contrôler le personnage de l'autre. Nous avons donc dû faire des modifications. Pour chaque personnage, nous avons rajouté une condition dans le code : si le personnage n'appartient pas au joueur qui joue (donc à son adversaire) il est désactivé uniquement du côté de ce joueur. Grâce à ce système, chacun peut contrôler librement son personnage.

Nous avons aussi dû nous occuper de la déconnexion des joueurs. Pour ce faire, lorsqu'un des joueurs (qu'il ait créé la room ou pas) se déconnecte, alors l'autre joueur est déconnecté et la room est détruite.

Physique du jeu

5.1 Les déplacements

La partie déplacement comprend les mouvements de base: avancer, reculer, tourner et sauter ainsi que les jeux et quelques interactions basiques telles que récupérer et poser des objets.

5.1.1 Mouvements de base

Après avoir consulté la documentation d'Unity et des tutoriels sur Youtube. Nous avons dû faire un choix entre faire les mouvements avec un character controller ou un rigidbody. Ce sont deux outils par défaut d'Unity qui permettent une implémentation facile des déplacements. Nous allons rapidement vous présenter quelques avantages de ces deux méthodes.

- **Character controller** : très facile de gérer les pentes, empêche le joueur d'être bloqué dans un mur et il n'est pas très dur de rendre les mouvements fluides.
- **Rigidbody** : gestion de la gravité intégrée et interaction facile avec différents objets physiques.

Notre choix c'est finalement porté sur le character controller car il nous semblait plus facile à mettre en place et le fait que le joueur ne se retrouve jamais coincé dans un mur nous semblait essentiel.

Avant de s'attaquer directement aux déplacements. Nous nous sommes occupés de faire un script nous permettant de gérer la sensibilité de la

souris, ainsi que le curseur ne soit pas visible et que la caméra ne puisse pas se déplacer à plus de 180° de la position actuelle du joueur. Cette étape était essentielle à faire rapidement pour nous permettre de pouvoir tester correctement notre code par la suite.

Nous avions maintenant toutes les cartes en main pour commencer le travail efficacement. La gestion des déplacements de base fut facile à mettre en place grâce à la méthode "Input.GetAxis" directement intégré dans Unity, qui permet de récupérer la valeur d'un axe virtuel, très pratique pour savoir si le joueur est en train d'avancer, de tourner etc. Le deuxième très utile est la structure Vector3, qui est très utile pour réaliser des actions sur 3 axes différents.

Pour s'occuper du saut du joueur, nous avons dû simuler la gravité dans notre jeu. Le fonctionnement est très simple. Il consiste à placer un objet vide en bas de notre joueur qui nous permet de vérifier si le joueur est bien en contact avec le sol. S'il ne l'est pas, il suffit alors d'appliquer la gravité sur notre joueur jusqu'à ce qu'il soit de nouveau en contact avec le sol.

5.1.2 Interactions de base avec un objet

Pour la mise en place des interactions entre un objet et le joueur, nous avions au début utilisé simplement des rigidbody et nous prenions l'objet le plus proche du joueur. Cependant cette façon de faire, nous a très vite bloqué lorsque nous voulions saisir un objet qui est derrière un autre.

Pour interagir avec un objet le joueur devra le viser et être à la bonne distance de l'objet puis appuyer sur la touche "E". Pour vérifier ces trois conditions, et en même temps nous libérer du problème du rigidbody, nous avons utilisé une méthode d'Unity : le Raycast. Cette méthode envoie un rayon (invisible aux yeux du joueur) à partir d'un point donné, dans une direction donnée et sur une longueur donnée. Une fois le rayon envoyé, on peut récupérer dans une variable toutes les informations le concernant, notamment s'il a rencontré un objet et si oui lequel. Il suffit alors de vérifier si on peut interagir avec cet objet. Si c'est le cas et que le joueur appuie sur la touche E alors l'interaction se lance.

Grâce à ce système, il devient possible d'ajouter n'importe quel type d'interaction. Nous pouvons donc attraper et poser un objet ou lancer une interaction plus complexe. Toutes les interactions sont décrites dans la partie "Mise en place des énigmes et missions".

Game Design

6.1 Interface utilisateur

Tout d'abord à l'aide de tutoriels et de la documentation d'Unity, Nous avons pu faire le menu de démarrage du jeu ce qui nous a permis d'avoir une idée des différents choix qu'aura le joueur aux lancements du jeu. Pour ce qui est du fond d'écran nous avons choisi cette image car nous trouvions qu'elle représentait bien le côté réflexion du jeu avec les neurones présents et le côté compétitif par le biais de l'athlète.

- **Options** : Un menu options, dans lequel le joueur peut modifier certains paramètres (qualité, résolution)



- **Menu pause** : En appuyant sur la touche "escape" quand le joueur est en partie, il accède à un menu lui permettant de modifier ses options ou encore de quitter la partie. Nous avons aussi résolu un problème qui était que la souris apparaissait mais lorsqu'on la bougeait, la caméra du joueur bougeait aussi, maintenant la souris peut bouger dans le menu sans déplacer la caméra du joueur.



- **Surlignement :** Afin d'améliorer l'expérience de jeu nous avons décidé de rajouter un curseur animé aux joueurs afin qu'ils puissent plus précisément interagir avec l'environnement. Grâce à l'aide de l'asset QuickOutline, des mêmes raycast utilisés dans la physique du jeu et d'un tag permettant de savoir si le joueur peut interagir avec l'objet, nous avons fait en sorte que lorsqu'un joueur regarde un objet avec lequel il peut interagir celui-ci se surligne. Cependant ajouter cette fonctionnalité nous a posé beaucoup de difficultés. Notamment le fait que le raycast passe à travers les murs. Le surlignement des objets se voyait donc à travers les murs. De plus au début nous utilisions une autre méthode qui consistait à changer le matériel de l'objet. Cependant avec celle-ci tous les objets étaient surlignés dès le début de la partie et il fallait les pointer au moins une fois chacun pour que cela fonctionne. Tous ces problèmes nous ont amené à utiliser l'asset QuickOutline.

Voici le résultat final:



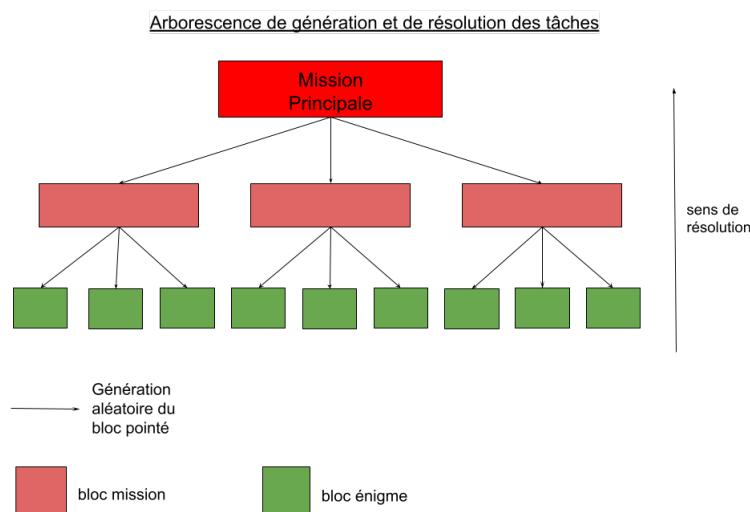
6.2 La modélisation 3D

A l'aide de plusieurs assets fournis dans l'unity assets store, nous avons pu créer une salle d'énigmes dans l'univers médiéval car il s'agissait là pour nous d'un thème qui correspondait parfaitement à l'ambiance de jeu que nous imaginions. Grâce à cela nous avons trouvé plusieurs objets correspondant au thème nous permettant de réaliser nos énigmes, par exemple différents tableaux ou encore le jeu d'échecs et le coffre en bois qui servent tous les trois à la réalisation d'éénigmes primordiales.

Algorithme de résolution

7.1 Principie

Pour terminer le jeu, le joueur doit résoudre toutes les énigmes et missions de la salle. Cependant, pour ajouter un peu de difficulté, les tâches sont réparties sous la forme d'un arbre avec en racine la dernière mission à résoudre et en feuille les premières énigmes à résoudre. Chaque tâche, lorsqu'elle est réussie, donne accès à un élément pour résoudre sa tâche parent. Grâce à ce principe, il est possible de remonter l'arbre de bas en haut jusqu'à finir le jeu. Vous trouverez ci-dessous un schéma d'arbre hypothétique.



Ici la mission principale est la dernière à être résolue alors que les énigmes seront les premières.

7.2 Fonctionnement

Les énigmes sont des composantes du jeu pouvant être résolues à tout moment. Elles constituent généralement un casse-tête pouvant prendre différentes formes (Ex : trouver le bon mouvement à faire sur un plateau d'échecs/trouver le bon objet à déplacer dans la pièce). Une fois résolue, l'énigme donne accès à un indice permettant de résoudre une mission interne.

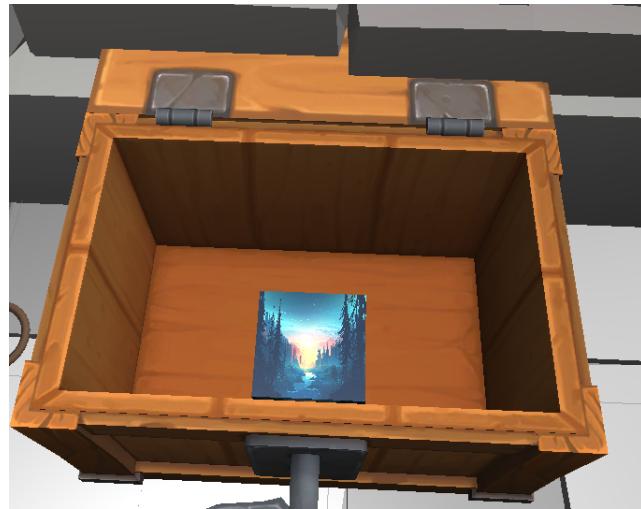
Les missions sont des composantes qui peuvent être résolues uniquement si tous les indices les concernant ont été récoltés. Une fois résolue, une mission peut agir de deux manières différentes en fonction de son importance.

- **Mission Principale :** Lorsque la mission est résolue, elle met fin à la partie. Le premier joueur à la résoudre gagne.
- **Mission Interne :** Lorsque la mission est résolue, elle donne accès à un indice pour résoudre la mission principale.

7.3 Implémentation

Pour Implémenter notre algorithme, nous avons tout d'abord créé un script qui permet de contrôler l'avancement du joueur dans l'arbre. Pour ce faire, il contient une fonction, pouvant être utilisée n'importe où, pour indiquer au script si une tâche a été finie. Lorsque cette fonction est appelée le script sait exactement quelle énigme est finie et quel indice elle donne.

Avec ces informations, le script peut désactiver la tâche pour empêcher le joueur de la faire en boucle et placer l'indice dans le coffre prévu à cet effet. Une fois l'indice placé dans le coffre, le joueur peut l'ouvrir et récupérer son indice.



Une fois que l'un des deux joueurs termine toutes ses énigmes, Photon envoie un évènement à l'autre joueur pour lui indiquer la fin de la partie. Une fois l'évènement envoyé un écran de victoire apparaît pour le joueur ayant gagné et pour celui ayant perdu, un écran de défaite.



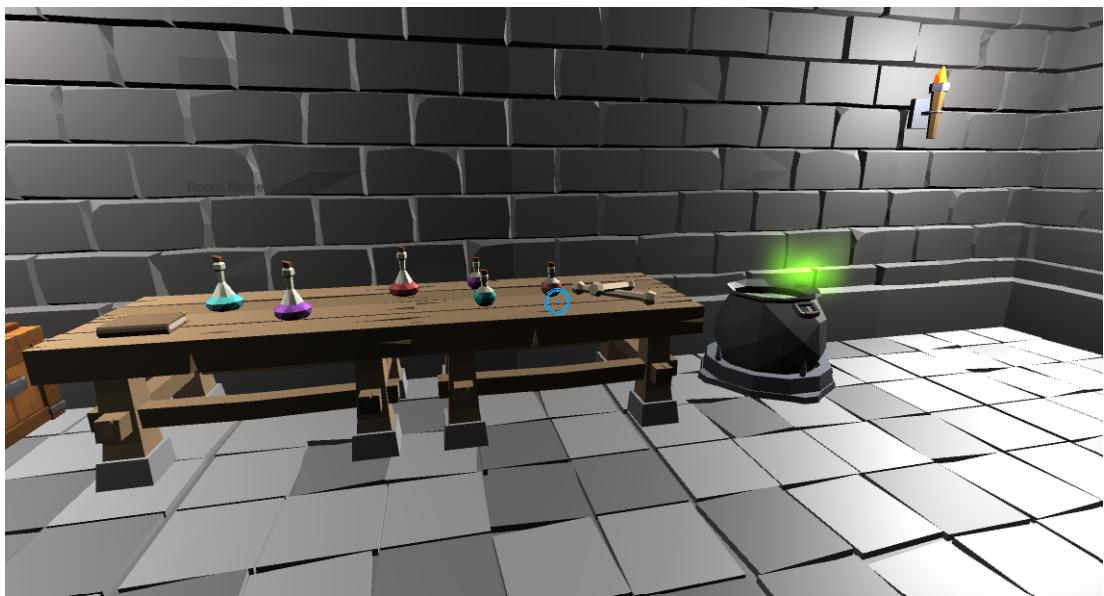
Le bouton "QUIT !" permet, comme son nom l'indique de quitter la partie. Pour rejouer, il faudra recréer une nouvelle room ou en rejoindre une déjà existante.

Mise en places des énigmes et missions

8.1 L'alchimie

Principe: Le but du joueur est de choisir entre différentes potions positionnées sur une table et de les mettre dans le bon ordre dans un chaudron. Si le joueur se trompe dans la réalisation de cette énigme toutes les potions seront éjectées du chaudron et il devra tout recommencer.

Fonctionnement: Chaque potion ainsi que le chaudron possèdent le layer Interaction qui permet l'interaction entre le joueur et les GameObject. Pour vérifier que le joueur insère les potions dans l'ordre, les potions possèdent chacune leur propre tag qui se découpe en deux parties (Exemple de tag: Potion, potion1). Cela nous permet de connaître le type de l'objet et d'avoir accès à son numéro associé. Pour renvoyer les potions du chaudron lorsque le joueur se trompe, nous utilisons simplement un "Rigidbody.AddForce".



8.2 L'échiquier

Principe : Lorsque le joueur interagit avec le plateau d'échecs, il va avoir une interface 2D d'une partie d'échecs déjà commencée qui va apparaître à l'écran où il suffit de faire 2 mouvements pour mettre l'autre équipe en échec et mat. Ce plateau de jeu se génère au début de la partie. Si le joueur ne fait pas le bon mouvement, la pièce déplacée revient à son emplacement initial. Une fois le roi adverse mit en échec et mat, lénigme enclenche un autre évènement.

Fonctionnement : Au lancement de la partie, un plateau de jeu est sélectionné parmi une base de données de plateau de façon aléatoire. La façon dont sont enregistrés les plateaux est assez simple, c'est un tableau de 9 lignes et 8 colonnes. Les 8 premières lignes représentent l'emplacement des différentes pièces, par exemple "BT1" représente la pièce "tour noire n°1". La dernière ligne permet de savoir quels sont les mouvements à faire par le joueur, puis le troisième élément permet au programme de savoir quoi faire une fois que le joueur a fait le premier mouvement. Ici "WQ,0,6" veut dire que le premier mouvement que le joueur doit effectuer est le déplacement de la dame blanche sur la case (0,6), et "0,4,0,6" représente

d'abord les coordonnées de la pièce à bouger (0,4), puis vers où la déplacer (0,6), donc la tour noire n°1 va manger la dame blanche.

```
private string[,] enigmeBoard1 =
{
    {"", "", "", "BQ", "BT1", "", "", "BR" },
    {"BP1", "BF1", "BP3", "BP4", "", "", "BP5", "BP6" },
    {"", "BP2", "", "", "", "", "", "WC1" },
    {"", "", "", "WQ", "", "", "BF2", "" },
    {"", "", "WP1", "", "WP2", "", "", "" },
    {"WP3", "", "", "", "", "", "", "" },
    {"WP4", "WP5", "", "", "WP6", "WP7", "WP8" },
    {"WT1", "", "", "", "", "WT2", "WR", "" },
    {"WQ,0,6", "WC1,1,5", "0,4,0,6", "", "", "", "", ""},
};

void Start()
{
    board.Create();
    pieceManager.Setup(board, enigmeBoard1);
}
```



Ici la séquence de jeu donne : Dame Blanche à côté du roi noir, puis tour noire mange la dame, puis cavalier met en échec et mat le roi en se plaçant en (1,5).

La base de données des plateaux d'échecs a été remplie grâce à ce site internet : http://chess.strategy.pagesperso-orange.fr/Palview/exercices/ex_mat_en_2_coups_V1.htm

8.3 Enigme visuelle et calcul mental

Principe: Lorsque le joueur va interagir avec un livre posé sur la table, une interface va alors apparaître et le joueur pourra choisir entre deux jeux. Le premier jeu comporte cinq questions de calcul mental et le second jeu contient deux charades. Le calcul mental est sous forme de QCM avec quatre possibilités pour chaque question et chaque charade, le joueur doit saisir sa réponse dans un champ de saisie. Si le joueur se trompe, il devra alors recommencer le jeu depuis le début.

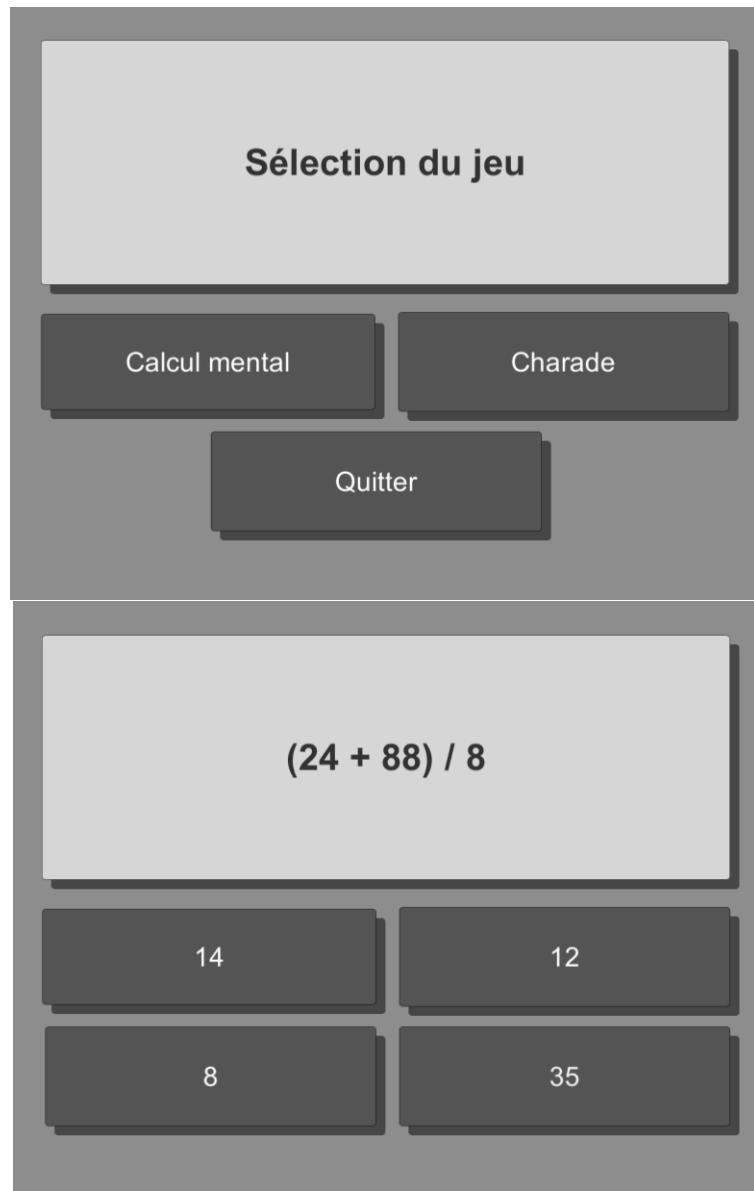
Fonctionnement: Le fonctionnement de l'énigme se décompose en deux parties majeures. La première est une succession de canvas, il s'agit donc de la partie visible de l'énigme. La deuxième partie est un "empty object", dans lequel est placé un script permettant la gestion des questions, qui sont

toutes stockées dans un tableau. Nous allons donc détailler chacune des parties.

- **Première partie :** La première fenêtre visible par l'utilisateur est la fenêtre principale. Elle permet au joueur de sélectionner le jeu de son choix et de simplement quitter la partie. Lorsque l'utilisateur choisit son jeu. Toutes les questions qui ne sont pas utiles sont alors supprimées du tableau où elles sont stockées.

Si le joueur choisit le calcul mental, un nouveau canvas va être généré. La question va être affichée au milieu de l'écran et sur chaque bouton va être écrit une réponse possible. Ainsi le score du joueur va être enregistré et en fonction de cela, deux choses peuvent se produire. Si le score du joueur est égal au nombre total de questions, le joueur a alors réussi le jeu et une fenêtre le félicitant apparaît. Dans le cas contraire la page qui apparaît indique le score du joueur et lui propose soit de quitter la partie soit de recommencer. (Si le joueur choisi au début le jeu des charades, la page générée est légèrement différente car elle possède un champ de saisie au lieu des boutons).

- **Deuxième partie :** Cette partie permet la génération et le choix des questions qui seront posées au joueur. Nous avons créé une classe (appelé QuestionAndAnswer) qui permet de stocker: la question, les quatre réponses possibles ainsi que la réponse de la charade si nécessaire. Ainsi toutes les questions sont dans un tableau de QuestionAndAnswer. Dans l'interface d'Unity, nous avons changé les questions et les réponses très rapidement. Nous avons inséré un système pour proposer les questions au joueur dans un ordre aléatoire, ce qui permettra de rendre le jeu plus vivant dans le cas où le joueur se serait déjà trompé.



8.4 Bibliothèque

Principe: Le joueur se retrouve face à une bibliothèque dans laquelle des livres ne sont pas bien rangés. En cliquant sur un livre on pourra inverser sa position avec le livre de gauche. Quand tous les livres seront triés, le joueur validera l'énigme.

Fonctionnement: Pour la mise en place de lénigme, nous avons eu besoin de Raycast et d'un tableau stockant tous les livres à déplacer. En premier temps, le raycast va détecter le livre qui se trouve en face du joueur. Puis si l'utilisateur appuie sur la touche E, on récupère l'élément positionné avant dans le tableau et on inverse leur position, sans oublier d'actualiser la position des éléments dans le tableau. Pour pouvoir vérifier que lénigme soit validée, nous avons appelé chaque livre avec un nombre (bien sur, tous les nombres sont distincts). Si le tableau se retrouve trié en ordre décroissant, lénigme est validée.



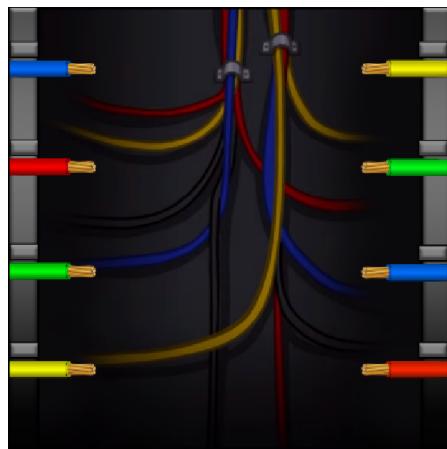
8.5 Les câbles

Principe: Cette énigme est inspirée de la mission des câbles du jeu Among Us. Lorsque le joueur va interagir avec un objet spécifique, une interface 2D va se générer. Dans laquelle la partie gauche est un début de câble et la partie droite est une sorte de prise femelle où le joueur devra insérer le bon câble dans la bonne prise en fonction de la couleur de ceux-ci. Dès qu'un câble est bien branché, une lumière au-dessus du câble s'allume. Lorsque toutes les lumières sont allumées, le joueur gagne la partie.

Fonctionnement: Chaque élément du jeu est un sprite 2D. Pour permettre au câble de s'allonger pour les amener sur la prise femelle, il s'agit d'aller dans le sprite editor d'Unity. Par la suite le déplacement d'élément se fait

très facilement grâce à la méthode d'Unity: OnMouseDrag. Nous avons aussi décidé de rendre l'ordre des câbles aléatoire pour qu'il puisse y avoir un léger changement lorsque le joueur veut rejouer l'éénigme et donc augmenter la durée de vie de notre jeu.

Remarque: Malheureusement, nous avons décidé de ne pas ajouter cette éénigme à notre jeu. En effet, l'éénigme marchait parfaitement en local mais quand nous voulions l'implémenter dans le multijoueur, le fait que tous les éléments soient des sprites posaient problème.



8.6 Les horloges

Principe: Cette éénigme n'est accessible qu'après avoir débloqué la seconde partie de la salle, son principe est assez simple, dans la salle principale il y a une horloge affichant une heure qui avance au fur et à mesure de la partie. Le but est d'interagir avec la deuxième horloge dans la petite salle pour pouvoir faire apparaître une interface et ainsi changer l'heure de cette dernière, et faire en sorte que les 2 horloges soient sur le même horaire à l'instant où l'on termine le changement d'heure.



Fonctionnement: Cette énigme se compose en 2 parties, la première consiste à simplement faire fonctionner une horloge, et la deuxième à pouvoir interagir avec l'horloge pour changer l'heure. Donc pour la première partie, nous avons un modèle d'horloge avec 2 aiguilles ainsi qu'un script. Ce script est assez simple, on initialise l'heure et les minutes à des valeurs aléatoires, puis on fait avancer le temps 15 fois plus vite (la valeur des minutes augmente toutes les 4 secondes). Et puis à la fin de ce script on traduit ces valeurs en degré pour faire pivoter les aiguilles.



Pour la seconde partie, un second modèle d'horloge est placé, il est surligné lorsque l'on survole l'objet pour indiquer que l'on peut interagir avec. Les

aiguilles de cette horloge ne bougent pas en fonction du temps, mais si on interagit avec, une interface 2D apparaît avec un cadran d'horloge et 2 aiguilles. D'abord l'aiguille des minutes tourne sur son axe et suit de sa pointe la souris, dès que l'on fait un clic gauche, c'est l'aiguille des heures qui fait la même chose, puis lorsque l'on clic à nouveau, l'interface disparaît et les aiguilles de l'horloge 3D se positionnent exactement comme celles de l'interface. Puis à la fin du programme, on vérifie si les aiguilles ont les mêmes rotations que celles de l'horloge qui bouge, si oui, l'éénigme est résolue, on empêche le surlignement et le fait de pouvoir relancer l'interface. Pour pouvoir accéder aux aiguilles nous avons un tableau de 6 éléments avec, les aiguilles de l'interface, puis de celle-ci, puis de celle qui tourne toute seule. Un paramètre "finished" permet de savoir si l'éénigme a été résolue.

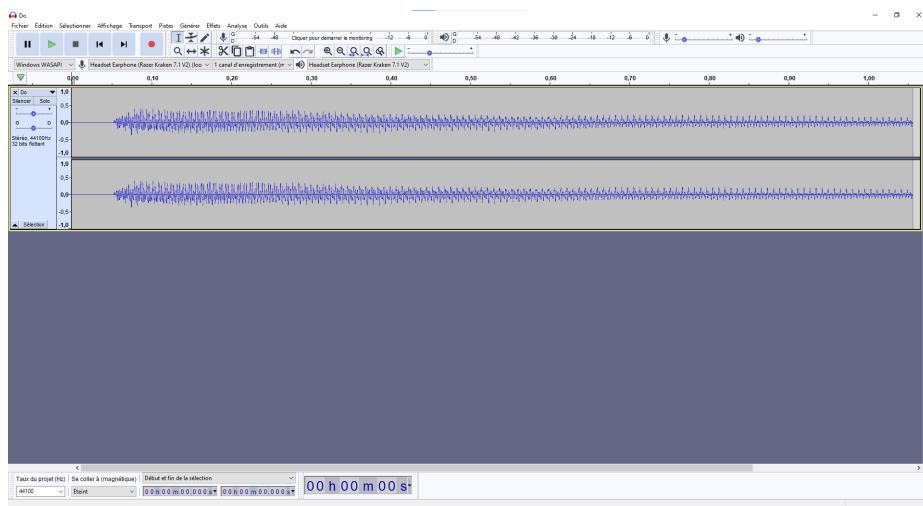
8.7 Enigme sonore

Principe: Le joueur devra trouver des lanternes qu'il mettra sur un piller, suite à cela une musique qui aura été générée aléatoirement au début de la partie constituée d'une suite de notes se jouera. Le joueur devra alors interagir avec 6 leviers qui, pour chacun, en fonction de leur position effectueront une note différente. Ensuite, le joueur validera la position des leviers en appuyant sur un bouton. Si la position des leviers n'est pas bonne, le joueur pourra réessayer. Cependant, le nombre de fois qu'il pourra écouter la musique originale sera limitée, au-delà, il devra trouver de manière aléatoire, ce qui lui fera perdre beaucoup de temps.



Fonctionnement: Afin de pouvoir ajouter des sons à notre jeu nous avons dû créer une classe son et c'est grâce à celle-ci que nous avons pu réaliser l'énigme des leviers. Nous sommes allé sur un site permettant de jouer du piano en ligne et à l'aide du logiciel Audacity nous avons fait une douzaine de piste audio comprenant chacune une note de piano différente. Au début de la partie se génère une liste de six éléments composée aléatoirement de zéro et de un qui correspondent à la position finale dans laquelle les leviers doivent être positionnés. Cette liste est assignée au pilier sur lequel le joueur devra interagir avec la lanterne en main. L'une des difficultés rencontrées a été d'effectuer un délai entre les différents sons pour que le joueur puisse distinguer clairement le son à reproduire pour cela nous avons utilisé la fonction "Invoke" qui nous a ainsi permis de mettre en place ce délai.

Exemple d'une note enregistrer sur Audacity:



8.8 Éléments à ouvrir

Principe: Ce n'est pas une énigme, mais c'est tout de même un élément important du jeu car il s'agit du coffre et de la porte qui se trouvent dans la salle. Pour la porte, il suffit d'avoir la clef pour déverrouiller l'accès à l'autre salle en ouvrant la porte. Le coffre lui permet au joueur de récupérer les objets donnés à la fin des énigmes réussies (on peut en avoir

l'illustration [ici](#)).



Fonctionnement: Pour la porte, son fonctionnement est simple, lorsque le joueur interagit avec la porte, on vérifie s'il a un objet à la main et si la porte est fermée, si ces deux conditions sont vraies, on vérifie que ça soit bien la clef qui permet de l'ouvrir, on déclenche l'animation de l'ouverture de la porte, on empêche que la porte puisse se rouvrir grâce à un booléen et on empêche aussi le fait qu'elle puisse être surlignée.

Au départ le coffre avait lui aussi une clef et ne pouvait s'ouvrir qu'une seule fois, un peu comme la porte, mais finalement nous avons décidé de l'utiliser comme un lieu pour récupérer les objets des énigmes. Son fonctionnement est le suivant, à chaque objet donné, le joueur doit ouvrir le coffre et une animation se joue, dès qu'une autre énigme est terminée, on joue une animation qui le ferme.

8.9 Les peintures

Principe: Dans notre salle, quatre cadres sont disposés sur les murs, chaque cadre possède un tableau. Dans cette énigme, les tableaux ne sont pas mis dans les bons cadres. Le but du joueur sera d'échanger les tableaux de place pour les faire correspondre aux bons cadres. Pour l'aider, le joueur possède un indice : sur chaque cadre est inscrit, en petit, le nom du tableau

à mettre à cet emplacement. Ce sera au joueur de comprendre cet indice pour replacer les tableaux correctement.



Fonctionnement: Le fonctionnement de cette énigme est assez simple.

Lorsque le joueur veut interagir avec un tableau plusieurs configurations sont possibles :

- Le cadre possède un tableau et le joueur a les mains vides : le joueur récupère le tableau et le cadre est alors vide.
- Le cadre est vide et le joueur tient un tableau : le cadre récupère le tableau en question et le joueur a de nouveau les mains vides.
- Le joueur tient un objet qui n'est pas un tableau : il ne se passe rien.

Lorsque tous les tableaux sont à la bonne place, l'énigme est terminée et elle se désactive, il devient donc impossible d'interagir avec les cadres.

8.10 Le puzzle

Principe: Un puzzle est disposé dans la salle et celui-ci possède quatre pièces manquantes. Pour obtenir ces pièces, le joueur devra résoudre quatre énigmes, une par pièce. Lorsque le joueur interagit avec le puzzle une interface apparaît à l'écran. Elle est composée de deux parties : en haut, l'image du puzzle avec les pièces manquantes et en bas, des icônes sur lesquelles viendront se placer les pièces retrouvées.



Fonctionnement: Pour faire fonctionner cette mission, nous avons d'abord récupéré une image que nous avons ensuite découpée en cinquante sous images. Grâce à ce découpage, il suffit de désactiver les quatre petites images. Chacune de ces quatre petites images est donnée en récompense de quatre missions.

Au moment où le joueur interagit avec le puzzle présent dans la scène, il verra apparaître à l'écran une image similaire à celle ci-dessus. Il peut quitter cette interface en appuyant sur la touche "escape" et la réactiver en interagissant de nouveau avec le puzzle.

Lorsqu'une des images est trouvée par le joueur, elle est activée par un script et apparaît dans un des quatre emplacements en dessous du puzzle. Le joueur peut alors la déplacer pour la mettre au bon endroit dans le puzzle. Une fois le puzzle complet, l'énigme se désactive.

Site Web

<https://synapsesite.github.io/>

9.1 Outil utilisé

C'est Thomas qui s'est chargé de la mise en place du site.

HUGO

HUGO est un framework open source qui a pour but de faciliter la création de site web static.

Pourquoi avoir utilisé ce logiciel ?

HUGO nous a semblé rapidement extrêmement intéressant. En effet, ce framework a très peu de liens avec des alternatives telles que WordPress ou bien Wix car il demande de véritables notions de code (HTML5, CSS3 et javascript) pour réaliser le site web.

Une fonctionnalité d'HUGO qui nous a fortement intéressé était la gestion du "responsive" qui est très facilité. De plus, travailler avec HUGO est très agréable car lorsque nous travaillons en local pour construire le site, nous pouvons le mettre dans un des serveurs d'HUGO qui permet de voir en temps réel sans même recharger la page l'avancement de notre projet.

9.2 Mise en page

Notre site est constitué de quatre fenêtres différentes:

- La première fenêtre, qui est aussi la principale, est constituée d'une description de notre jeu. De plus, on y trouve un bouton pour télécharger le jeu sous Windows ou sous Linux. On y trouve aussi une courte vidéo de notre jeu, pour permettre à l'utilisateur de comprendre qui est le groupe et quel est le type de jeu.
- La seconde fenêtre explique le fonctionnement détaillé de notre jeu, tel que les règles du jeu et le fonctionnement détaillé de la partie.
- La troisième fenêtre regroupe tous les liens qui peuvent être utiles au joueur tel qu'un lien vers notre GitHub, et avoir accès à tous nos rapports de soutenance et cahier des charges.

Chaque fenêtre possède les mêmes caractéristiques:

- **L'entête :** qui prend presque toute la place disponible sur l'écran de l'utilisateur. Il est constitué du titre de la fenêtre actuellement affichée, d'une barre de navigation permettant de se déplacer entre différentes fenêtres. Chaque fenêtre possède aussi une courte description pour que l'utilisateur puisse se repérer facilement dans notre site.
- **Corps de la page :** Regroupe toutes les informations importantes de la page telle que du texte, des liens, et des images.
- **Bas de page :** Il s'agit d'un bandeau noir identique en bas de chaque page du site. Il contient un lien vers notre GitHub ainsi que le nom de notre jeu.



Synapse en quelques clics:

Promotion: EPITA

19.99€ 0.00€

N'attends plus et rejoins la communauté [Synapse](#) dès maintenant !

Tu hésites encore?

A screenshot of the "Fonctionnement du jeu" (How the game works) section of the Synapse Project page. It features a black and white photograph of a chessboard with pieces. Overlaid on the image is the text "Fonctionnement du jeu". Below the image, there is a paragraph of text explaining the game mechanics: "Synapse est un jeu qui met en compétition deux joueurs. Chacun apparaît dans une salle d'énigmes générées aléatoirement. La salle est la même pour les deux joueurs mais ils n'ont aucun moyen de se voir. Le premier qui complète ses énigmes gagne la partie !" At the top of the section, it says "Synapse Project" and "Fonctionnement du jeu Annexes" with a small logo.

9.3 Hébergement

Pour l'hébergement de notre site web, nous avons décidé d'utiliser GitHub Pages, pour sa facilité d'utilisation et son tarif très intéressant puisque gratuit.

Le fonctionnement est très simple, nous avons dû créer un nouveau compte GitHub qui aurait le nom de notre site web sinon cela aurait créé des soucis au niveau de l'URL. Par la suite, nous avions juste à build notre projet et l'insérer dans notre nouveau répertoire GitHub Pages. Par la suite, nous avons juste dû faire quelques manipulations pour modifier les paramètres de notre site.

Trailer

Pour donner envie de jouer, nous avons décidé de réaliser un trailer qui pourrait aussi faire office de présentation de notre jeu.

Cependant, dans le groupe personne n'avait de connaissance en montage vidéo. C'est pour cela que nous nous sommes appuyés sur le logiciel Wondershare Filmora X, qui est un logiciel de montage vidéo, très facile à utiliser mais relativement complet.

Nous avons choisi que des vidéos de notre jeu, et nous avons décidé de montrer une petite partie de chaque énigme pour montrer l'étendue de notre jeu. De plus, sur chaque partie du trailer, nous avons ajouté un titre en bas à gauche de la vidéo pour expliquer le principe et les caractéristiques de notre jeu.

Pour finir, pour accompagner notre vidéo, nous avons ajouté un remix de la musique fortitude de Lance Conrad. Ce remix s'est directement bien intégré dans le trailer et les changements que nous avons dû faire sur la vidéo ont été que minime.

Difficultés rencontrées

11.1 Site web

Lors de la conception du site web, nous nous sommes retrouvés face à de nombreux problèmes que nous n'avions pas anticipés:

- **Hugo (logiciel)** : Le premier problème que nous avons rencontré est purement lié au logiciel. En effet, Hugo est principalement utilisé pour la conception de blog et non pour la création d'une page pour un jeu vidéo. Nous nous sommes donc très vite senti bridé par le logiciel qui ne répondait totalement à nos attentes (il cherchait en permanence à faire des liens entre les pages qu'il considérait comme des articles de blog).
- **Markdown** : La deuxième difficulté est liée au markdown. En effet, lorsque l'on travaille Hugo le langage le plus utilisé est le markdown et dès que l'on décide de "build" le site pour pouvoir l'héberger. Hugo transforme le markdown en HTML. Nous avons donc été obligé de beaucoup travailler sur la version prête à être hébergé, pour pouvoir bénéficier des fonctionnalités non disponibles avec markdown.

11.2 Physique du jeu

Lorsque nous avons travaillé dans les interactions joueur/ objet, nous avons au début simplement placé un RigidBody sur un objet et utilisé un script qui récupère l'objet le plus proche. Cependant, cette façon n'était pas viable puisque nous ne pouvions récupérer un objet placé derrière un autre. Nous

avons donc utilisé les Raycast, qui nous ont permis de saisir les objets de façon très précise.

11.3 Enigmes

Nous allons maintenant vous présenter différentes difficultés que nous avons rencontrées lors de la conception des énigmes de notre jeu.

- **Les sprites :** Il s'agit d'un problème que nous avons lors de la conception d'éénigme (par exemple: les câbles). Lorsque nous implémentions les énigmes en multijoueur, les canvas qui utilisaient des sprites n'apparaissaient pas sur l'écran du joueur. Ce problème, nous a énormément retardé dans la conception des énigmes et nous n'avons pas trouvé de moyen pour le résoudre.
- **La gestion des "transform" des horloges :** Ce problème était premièrement que nous n'arrivions pas à récupérer les transform des aiguilles des trois horloges à partir des objets de jeu des horloges, à chaque fois que nous utilisions la méthode "GetComponent" pour récupérer les transform des éléments des objets de jeu des horloges, le code cessait de continuer l'exécution. Pour pallier ce problème nous avons mis en place une liste répertoriant tous les objets de jeu de chaque aiguille.

Dans une seconde partie, le problème était que les transforms ne permettent pas d'avoir des angles en degré, mais plutôt des "quaternion", qui sont difficiles à comprendre, nous avons dans un premier temps tenté de les comprendre pour pouvoir comparer les angles des aiguilles des 2 horloges 3D, mais nous avons fini par trouver une autre manière, qui est de convertir en angles d'Euler avec une méthode, et d'ainsi pouvoir les comparer.

- **L'interface utilisateur avec le multijoueur :** Ce dernier n'était visible que lorsque l'on lance une partie à deux joueurs, en effet il posait deux problèmes, le premier était que lorsqu'un joueur souhaitait mettre pause, l'autre joueur ne pouvait plus bouger sa souris, cela venait de

la manière dont nous avions géré le blocage de la souris sur le menu pause. Le second était similaire, lorsque les deux joueurs souhaitaient faire la même énigme impliquant l'interface utilisateur en même temps, seul un des deux joueurs avait le contrôle de cette dernière. Par exemple s'ils lançaient l'énigme des horloges, seul un des deux joueurs avait la possibilité de bouger les aiguilles, et parfois même elles étaient incontrôlables.

- **L'énigme des peintures :** Lors du développement de l'énigme, les tableaux apparaissaient bien dans les cadres sur la version de développement mais dans la version build du projet les tableaux n'apparaissaient pas. Après de longues heures de déboggage, nous avons compris que ce problème était dû à une différence d'ordre d'exécution des scripts entre les deux versions. En effet, lorsque le jeu commence, Unity appelle toutes les fonctions "Start" du projet. Dans la version de travail, l'ordre d'appel était tel que tout fonctionnait correctement. Or dans la version build, l'ordre n'étant plus le même, les peintures n'apparaissaient pas. Pour corriger le bug, nous avons donc exécuté le code dans la fonction "Start" défaillante à un autre moment.

Ressenti personnel sur le projet

12.1 Thomas

D'un point de vue global, je suis très satisfait d'avoir pu réaliser mon premier jeu vidéo. Cela a dû, nous obliger à devoir faire preuve d'organisation et de discipline. J'ai la sensation d'avoir progressé et gagné de bonnes habitudes d'un point de vue programmation (rester bloqué moins longtemps sur un problème, lire la documentation...).

La seule remarque négative que je pourrais faire sur ce projet est notre choix de jeu. Bien que ce jeu me plût énormément lorsque nous l'avions choisi. Nous avons dû très vite travailler sur la conception d'énigmes, ce qui s'est montré répétitif et m'a donné la sensation de faire une multitude de petits jeux sans jamais avancer sur notre projet.

Je suis par ailleurs très content d'avoir pu travailler sur le site web. Cela a pu me permettre de varier davantage mes tâches et pouvoir faire deux choses totalement différentes en même temps.

12.2 Olivier

Le projet m'a permis de mieux me rendre compte des étapes de création d'un jeu vidéo et plus particulièrement des problèmes rencontrés lors de la réalisation de celui-ci. Surtout sur l'aspect multijoueur et des différents bugs rencontrés au fur et à mesure de notre progression. De plus le fait de le réaliser en groupe nous a permis à une plus petite échelle de se rendre compte du travail d'équipe nécessaire dans de tels projets. L'une des

difficultés a été de répartir les tâches pour faire en sorte que tout le monde puisse travailler sur le projet en même temps et ainsi être le plus efficace possible.

Globalement j'ai apprécié réaliser ce projet et dorénavant me sens plus à l'aise dans l'idée d'une future création d'un jeu.

12.3 Hugo

Mon ressenti sur ce projet est plus que positif car malgré les nombreux problèmes que nous avons rencontrés, je peux enfin voir au final que ces difficultés nous ont permis de nous améliorer, voire de découvrir certains domaines de la programmation. Et c'est aussi grâce à des situations pareilles que l'on a pu développer un certain sens de la communication et de l'entraide au sein d'un groupe travaillant un projet commun. J'ai aussi la sensation qu'avec ce projet de groupe, cela nous a permis de se partager les tâches de manière plus efficace.

Etant satisfait du travail que nous avons accompli, j'ai tout de même certains points que j'aurais aimé améliorer, par exemple je trouve que le son est une facette du jeu qui n'a pas été assez exploité. J'ai aussi un certain regret sur le manque de modélisation 3D, qui a été fait au profit de la création d'énigmes.

Mais je suis globalement satisfait du rendu final de notre jeu, ce projet m'a permis de travailler sur plusieurs domaines de la création d'un jeu vidéo, et je sens que la manière dont j'aborde un problème a changé tout au long du projet.

12.4 Léo

C'est la première fois je travaille sur un projet aussi important et durant une aussi longue période. Cela a augmenté ma capacité à travailler à plusieurs et m'a permis d'améliorer ma rigueur dans mon travail. Mon niveau en C# et mes connaissances d'Unity ont toutes les deux augmentées lors de la création de notre jeu.

Je trouve néanmoins que nous avons été trop ambitieux pour ce projet et j'aurais avec du recul, préféré faire un projet plus simple, mais pouvoir le terminer parfaitement et être sûr qu'il n'y ait pas le moindre bug.

Logiciels utilisés

Nous allons dans cette partie faire un rapide point des logiciels que nous avons utilisés pour réaliser ce projet en expliquant les choix qui nous ont poussé à les utiliser.

- **Unity** : C'est un moteur de jeu multiplate-forme développé par Unity Technologies. Gratuit et facile à prendre en main, c'est ce logiciel qui nous a permis de réaliser l'ensemble de notre jeu.
- **Rider et VisualStudio** : Ce sont des éditeurs de code que nous avons utilisé pour écrire les scripts nécessaires à notre jeu.
- **VisualStudio Code** : Il s'agit aussi d'un éditeur de code, cependant nous avons principalement utilisé VisualStudio Code pour pouvoir éditer notre site web.
- **Photon** : Photon est une plateforme de multijoueur et un moteur de mise en réseau. Il a été indispensable pour la mise en place du multijoueur de notre jeu.
- **Overleaf** : Il s'agit d'un éditeur latex en ligne, qui nous a permis de réaliser tous nos rendus écrits de façon propre.
- **Audacity** : C'est un éditeur audio que nous avons utilisé pour réaliser les bruits de certaines énigmes de notre jeu.
- **GitHub et Git** : GitHub et Git nous ont permis de partager et stocker notre code. Ces outils nous ont aussi permis d'être sûr de ne jamais perdre notre code.

- **Hugo** : Framework open source que nous avons utilisé pour le site web. Il propose un vaste choix de thèmes, ce qui facilite grandement la création du site web.
- **GitHub Pages** : Il s'agit d'un hébergeur que nous avons utilisé pour notre site web. Très facile à utiliser et totalement gratuit, parfait pour pouvoir héberger des sites web statiques.
- **Discord** : Discord est une messagerie digitale qui nous a permis de communiquer durant tout le projet: s'échanger des liens, faire des réunions et pouvoir s'aider facilement si une personne du groupe à des difficultés dans une tâche.
- **WonderShare Filmora X** : Filmora est un logiciel de montage vidéo qui nous a permis de réaliser le trailer de notre jeu. Facile à prendre en main mais également très complet. Cet outil nous a été d'une grande utilité.

Conclusion

Notre projet s'est déroulé comme nous l'avions souhaité, et c'est ainsi que l'aventure de Synapse prend fin. Nous sommes tous heureux d'avoir pu réaliser un jeu qui nous correspond en sachant qu'auparavant aucune personne du groupe n'avait des notions sur le moteur de jeu Unity, ou une quelconque autre expérience en création de jeu vidéo.

Durant ces quelques mois, nous avons appris à utiliser correctement Unity et améliorer nos compétences en C#. Nous avons également gagné en autonomie et appris à travailler en équipe, en planifiant notre travail et en faisant preuve de rigueur et de discipline.