

# Rapport de soutenance



Léo CAPMARTIN, Thomas CORBIERE  
Hugo SAINT GERMES, Olivier TALANE

3 Mars 2021

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Git et GitHub</b>	<b>3</b>
<b>3</b>	<b>Chronologie</b>	<b>4</b>
3.1	Le menu . . . . .	4
3.2	La modélisation 3D . . . . .	6
3.3	Les déplacements . . . . .	7
3.3.1	Mouvements de base . . . . .	8
3.3.2	Interactions de base avec un objet . . . . .	9
3.4	Le multijoueur . . . . .	9
3.4.1	La mise en place d'un serveur . . . . .	9
3.4.2	Installation d'un premier "jeu" multijoueur . . . . .	10
3.4.3	Finalisation du multijoueur . . . . .	11
3.5	Correction de bugs et finalisation . . . . .	13
<b>4</b>	<b>Avancement des Tâches</b>	<b>15</b>
<b>5</b>	<b>Tâches à venir</b>	<b>16</b>
5.1	Game Design . . . . .	16
5.2	Les énigmes . . . . .	16
5.3	L'intelligence artificielle . . . . .	17
5.4	Finitions . . . . .	18
5.4.1	Physique . . . . .	18
5.4.2	Interface . . . . .	18
5.4.3	multijoueur . . . . .	19
<b>6</b>	<b>Conclusion</b>	<b>20</b>

# Introduction

**Une fois notre cahier des charges validé nous avons commencé à nous organiser pour la réalisation de notre projet : Synapse.**

Nous nous sommes d'abord accordé sur les points à réaliser au plus vite et nous en avons conclus que le multijoueur et la physique du joueur était essentiels pour pouvoir implémenter la suite du projet. La partie Modélisation 3D et interface est aussi très importante pour la suite. En effet, sans celle-ci, il est impossible de travailler proprement sur le reste. Par exemple, sans niveau jouable nous ne pouvons pas tester la physique du joueur ou la connexion au niveau.

Notre rapport est basé sur 4 grands axes.

- *Travailler en commun : Git et GitHub*
- *Chronologie : Les tâches déjà effectuées*
- *Comparaison avec l'objectif fixé*
- *Le travail à venir*

# Git et GitHub

## **De quel moyen avons-nous mis en commun notre travail ?**

Notre priorité était de trouver un moyen efficace pour pouvoir partager, et mettre en commun notre travail. Nous nous sommes donc tourné vers l'application GitHub qui correspondait exactement à nos attentes.

Léo qui maîtrisait un peu plus le sujet s'est chargé de créer les différentes branches dont nous avions besoin (une branche pour la modélisation 3D, multijoueur etc). Nous nous sommes mis d'accord pour que chaque personne travaille sur sa branche. Lorsque le travail est fini et validé par tous les membres du groupe, nous mettons en commun grâce à l'outil de merge. La branche "dev" contient la version la plus à jour du code. De son côté, la branche "main", nous permet de stocker la version la plus stable du code.

Pour faciliter l'organisation du projet dans les différentes branches, nous utilisons l'application sublime merge, qui se révèle être un bon complément à GitHub, pour avoir un bon aperçut du travail de tout le groupe et d'avoir un moyen graphique pour ce que l'on décide de push.

## Chronologie

### 3.1 Le menu

Tout d'abord à l'aide de tutoriels et de la documentation d'Unity, Olivier a 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 choisis cette image car nous trouvions quelle représentait bien le coté de réflexions du jeux du aux neurones présentes et le cotés compétitif par le biais de l'athlète.



- **Sélection du mode de jeux :** Le joueur aura le choix entre un mode multijoueur, ce qui lui permettra de jouer en ligne en un contre un et un mode contre-la-montre, où il jouera en local avec un chronomètre permettant d'évaluer son temps.



- **Options** : Un menu options, dans lequel le joueur pourra modifier des paramètres à sa guise (volume,luminosité,...)



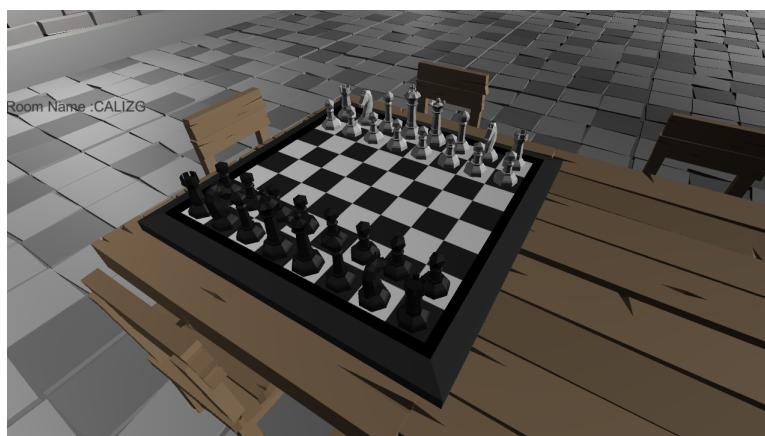
- **Menu pause** : En appuyant sur la touche echap quand le joueur est en partie, il accède à un menu lui permettant de modifier ses options ou encore de quitter la partie.



## 3.2 La modélisation 3D

A l'aide de plusieurs assets fournis dans l'unity assets store, Olivier a aussi pu créer une première salle d'énigme avec des éléments correspondant à celle-ci afin dans le futur de pouvoir interagir avec ces éléments.

- **Plateau d'échec** :Il permettra de mettre en place l'énigme dans laquelle, nous devrons bouger un unique pion afin de mettre en échec et mat l'équipe adverse.



- **La table d'alchimie** :Le joueurs devra interagir avec les fioles et les mettre dans le chaudron afin d'obtenir une potion spéciale lui permettant d'avancer.



- **Le reste du décor :**Dans lequel le joueur trouvera différents éléments(coffre,artefact,...).



Il était important pour nous, que le joueur est un espace suffisamment volumineux dans lequel évolué et qu'il soit composé de différentes pièces, afin que le joueur est vraiment cette impression de devoir chercher et que tous ne sois pas directement visible au premier coup d'oeil.

### 3.3 Les déplacements

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

Thomas c'est principalement occupé de la partie déplacement du joueur.  
Notre jeu ne possède que des déplacements à la première personne.

### 3.3.1 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 de 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 fait que le joueur ne se retrouve jamais coincé dans un mur nous semblait essentiel.

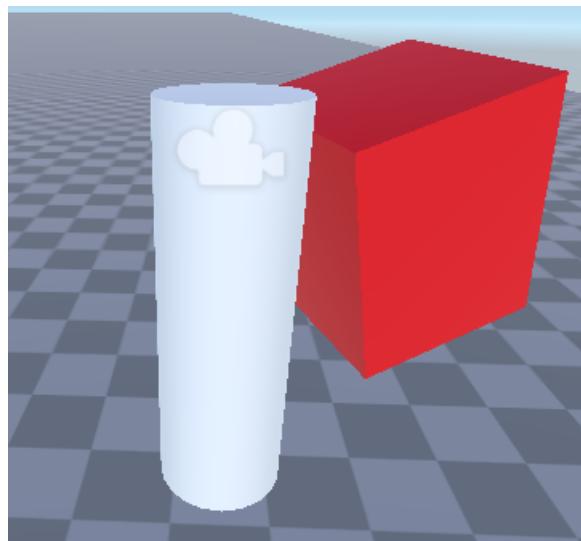
Avant de s'attaquer directement aux déplacements. Nous nous sommes occupé de faire un script nous permettant de gérer la sensibilité de la souris, ainsi que le curseur ne soit pas visible et que la caméra ne puisse pas se déplacer à plus de 180° de la position actuelle du joueur. Cette étape était essentielle 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 bases 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 simulé 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.

### 3.3.2 Interactions de base avec un objet

Pour mettre en place des interactions entre un objet et le joueur, nous avons utilisé un rigidbody, car comme expliqué précédemment c'est sans doute le moyen le plus simple et le plus pratique pour réalisé ce que nous voulions. Plus précisément, la possibilité de ramasser un objet, se déplacer avec, le poser et pouvoir le lancer. Ce qui nous sera très utile par la suite quand nous passerons à la partie des énigmes.



## 3.4 Le multijoueur

### 3.4.1 La mise en place d'un serveur

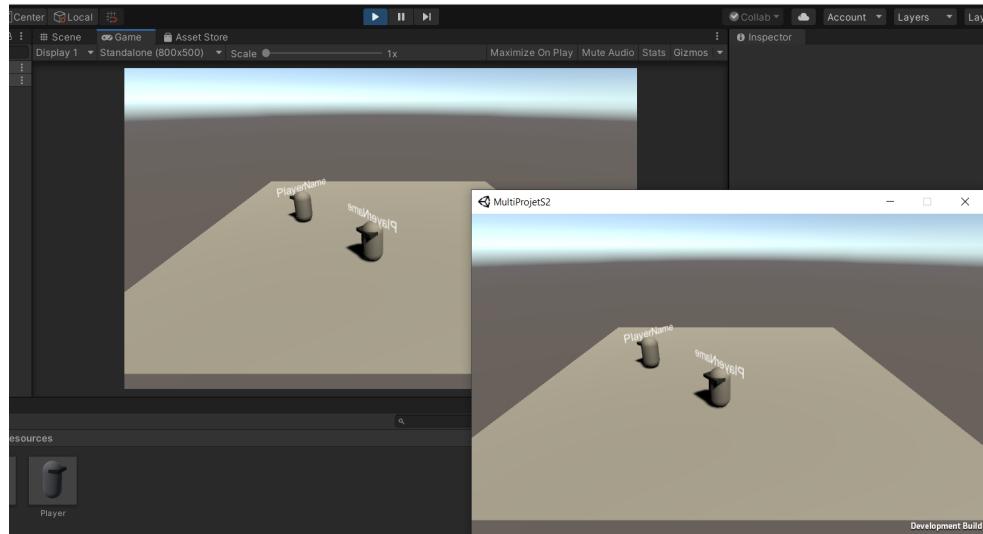
Pour le système de multijoueurs, nous avons fait le choix d'utiliser un asset de Unity : Photon (PUN 2), nous avons décidé de prendre ce dernier car il met à disposition un serveur en ligne gratuitement. Il nous permet

également d'avoir une facilité sur la gestion du lobby ou des rooms. 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 .



### 3.4.2 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 tout les joueurs. La prefab du player n'était pas celle de Thomas, mais plutôt un simple code pour faire bouger un objet (sans saut, sans caméra personnelle), cela a permis de se 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 prédéfinie dans le code en faisant apparaître le joueur sur l'un des spawn-point prédéfini (la réelle implémentation des rooms a été faite par Léo). C'est avec ce que l'on a appris sur ce projet que l'on a construit notre jeu.

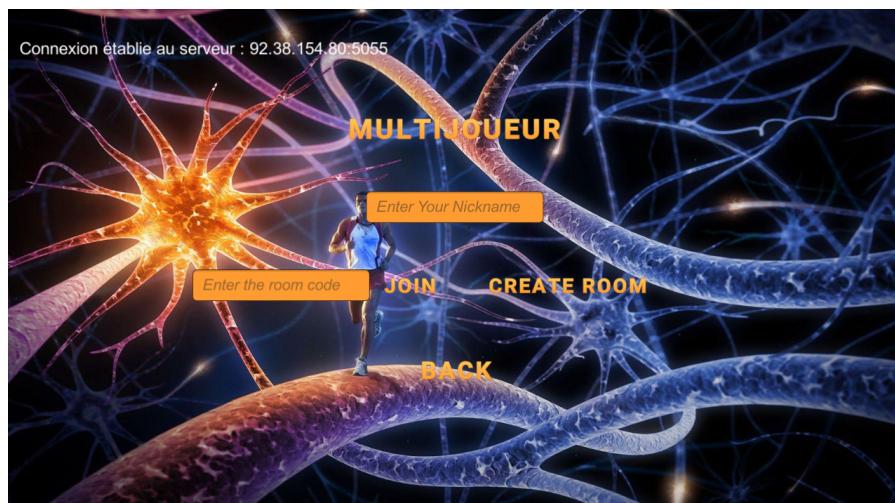


### 3.4.3 Finalisation du multijoueur

#### Création des rooms :

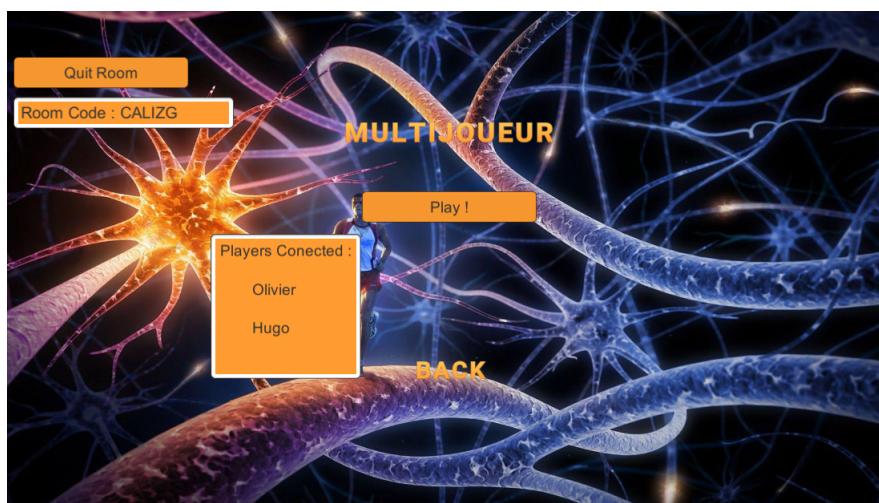
Une fois le serveur mis en place avec Photon 2, il fallait créer le système de room. Léo s'est chargé de créer celui-ci. Grâce à Photon, il est possible de créer des room assez facilement avec un nom personnalisé. Grâce à cette possibilité, nous avons pu implémenter un menu permettant de soit créer une room avec un nom de six lettres aléatoires, soit rejoindre une room déjà existante grâce à son nom.

Lorsque le joueur crée une room, il la rejoint automatiquement.

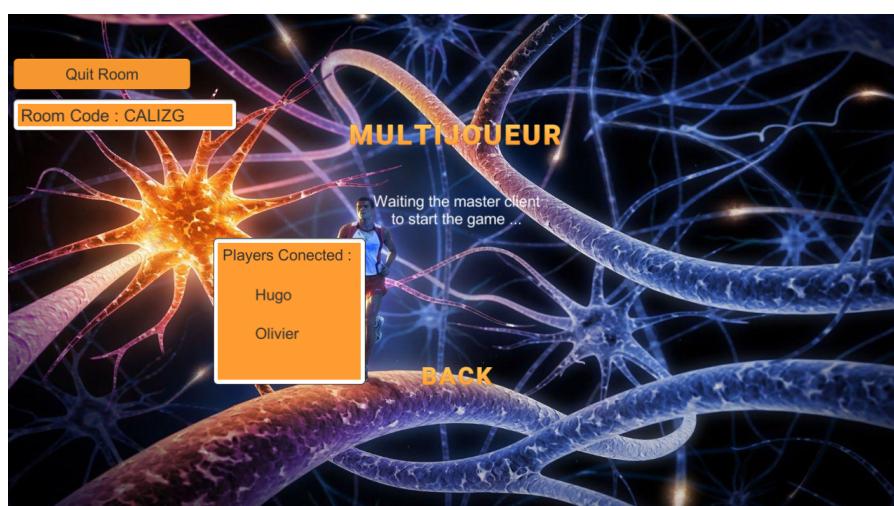


## Le lobby :

Une fois que le joueur entre dans une room, il est redirigé vers un autre menu sur lequel est affiché la liste des joueurs, le nom de la room, un bouton permettant de quitter la room et, si il a lui même créer la room, un bouton "Jouer". Ce bouton permet de lancer le jeu uniquement si 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.



Vue depuis le créateur de la room



Vue depuis le second joueur de la room

## En jeu :

Il fallait donc commencer par implémenter un vrai joueur avec des vrai contrôles et une caméra pour chaque joueur à la première personnes, ce joueur a été fait par Thomas, il fallait tout de même faire des modifications, pour que par exemple on ne puisse pas contrôler le joueur de l'autre personne, ou que l'on ne puisse pas contrôler l'orientation de la caméra de l'autre joueur.

Pour cela Thomas expliqua le fonctionnement de son code à Hugo pour qu'il puisse le modifier de sorte à ce que les joueurs apparaissent dans leurs points d'apparition respectifs en respectant les contraintes citées plus haut. En multijoueur, notre jeu ne peut se jouer qu'à deux, donc si un des joueur se déconnecte, l'autre est redirigé automatiquement vers le lobby ou le menu en fonction de si il est le créateur de la room ou pas.

## 3.5 Correction de bugs et finalisation

Au cours de notre développement, nous avons rencontré chacun différents bugs. Voici une liste non exhaustive des bugs les plus importants et de comment nous les avons corrigé.

- **personnage coincé dans le sol :** Bug : lors de la mise en place des mouvements du personnage, celui-ci avait très souvent la partie basse de son corps qui rentrait dans le sol. Ceci était très embêtant car empêché le joueur de sauter.

Solution: Changer la position de la variable qui s'occupe de la vitesse du joueur. En effet, nous changions la valeur de cette variable entre une condition qui vérifie que la vitesse n'est pas inférieur à 0 et l'actualisation de celle-ci sur le joueur.

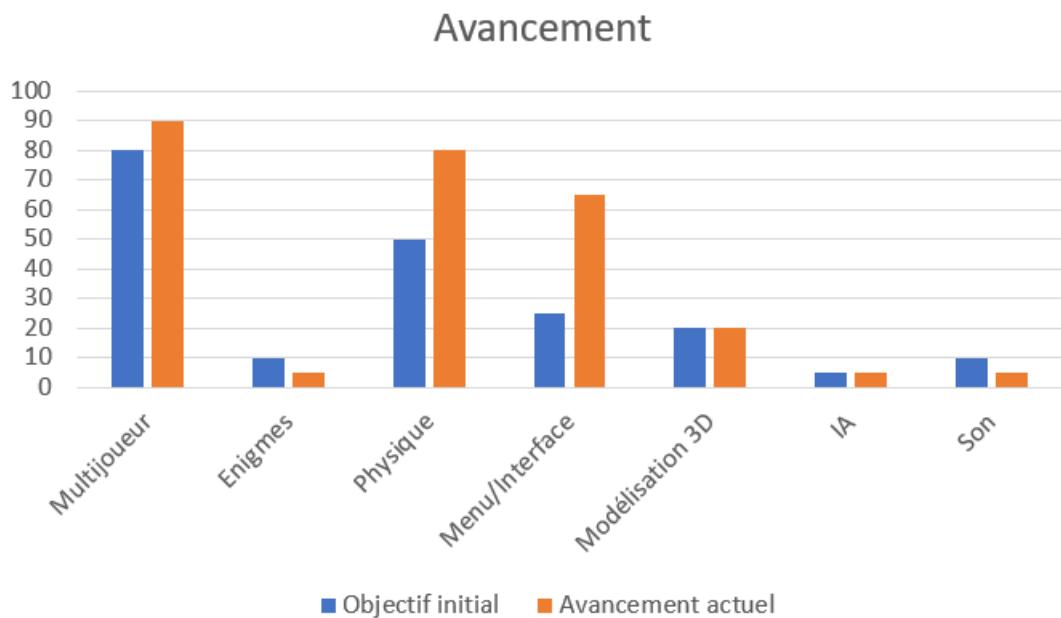
- **Bug de déconnexion :** Bug : Si le joueur créateur de la room se déconnecte alors, le deuxième joueur est lui aussi déconnecté. Pour mettre en place ce système, il existe la variable `isMasterCilent` permettant de tester quel joueur à les droits de la room. Par défaut, elle est attribuée au créateur de la room. Il suffit donc de tester si le joueur qui

se déconnecte possède la variable. Cependant, si le Master Client se déconnecte, sa variable est transférée à un autre joueur. Il est alors impossible de vérifier si le joueur qui se déconnecte est le Master Client. Solution : Il existe une fonction appelée par PUN lorsque le Master Client change de joueur. Pour régler le problème, il suffit donc de déconnecter tous les joueurs lorsque cette fonction est appelée.

- **Bug de lancement de la partie** : Bug : Lorsque les 2 joueurs rejoignent un même room ils sont dans le lobby et le bouton "jouer" s'affiche, mais nous avions eu des problèmes pour lancer la partie, si le joueur qui n'avait pas créé la room essayais de lancer la partie il y avait des soucis d'apparition des joueurs.  
Solution : Nous avons utilisé la variable évoquée dans le bug de déconnexion isMasterClient, pour que seul le joueur ayant créé la room puisse voir le bouton "jouer", et que si il y a bien 2 joueurs dans la room, la partie se lance, l'autre joueur voit seulement un message lui disant d'attendre que le Master Client lance la partie.
- **Contrôle des joueurs inversés** : Bug : Lorsque les joueurs apparaissaient sur la scène, le joueur 1 contrôlait le joueur 2 et inversement. Pendant longtemps on pensait que le problème venait d'un mauvaise gestion de Photon dans les scripts de déplacements et de gestion de la caméra, mais en réalité le problème était bien plus simple que ça, c'était la vision de la caméra qui était inversé. Ce problème vient du fait que la caméra que l'on voit à l'écran est la dernière qui apparaît, or le clone de l'autre joueur se fait après le notre, ce qui explique pourquoi on a sa vision.  
Solution : Il a tout simplement fallut vérifier si ce n'était pas notre joueur et si la caméra est désactivé, alors dans ce cas il faut la désactiver. Cette résolution de problème a permis de résoudre en même temps les messages d'alertes d'Unity indiquant qu'il y a plusieurs élément d'écoute dans la scène.

## Avancement des Tâches

Comme l'indique le histogramme ci-dessous, nous sommes globalement en avance sur nos attentes sur cette première section du projet. Nous avons par exemple une grande avance sur nos menus et notre physique.



Nous sommes donc en avance sur nos prévisions. Maintenant que la physique et le multijoueur est opérationnel, nous allons pouvoir nous concentrer sur la mise en place des énigmes et de l'intelligence artificielle.

## Tâches à venir

### 5.1 Game Design

Nous avons pour le moment une salle dans laquelle nous pouvons jouer. Celle-ci est dans un design plutôt médiéval. Nous aimerais si possible ajouter une ou deux salles différentes ce qui apportera plus de diversité dans notre jeu. L'ajout de ces salles ne changera en rien le gameplay ni les énigmes. Seul l'environnement dans lequel évolue les joueur sera modifier.

### 5.2 Les énigmes

Étant donné le fait que la partie physique et multijoueur soit suffisamment avancé pour nous permettre de faire toutes les actions nécessaire sans être gêné. Nous souhaiterons commencé à fortement s'investir dans la construction des énigmes.

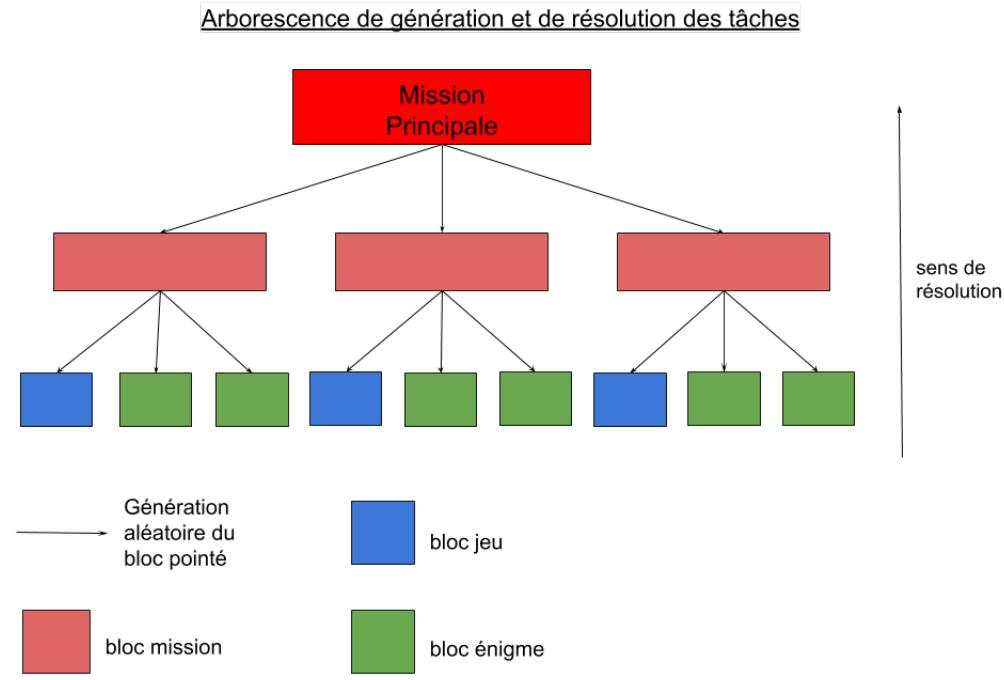
Voici une liste d'énigmes que nous aimerais faire pour la prochaine soutenance.

- **Éléments à ouvrir :** Un coffre, une boîte ou une porte seront à ouvrir avec un code. Les chiffres, les lettres ou les couleurs qui composent le code seront éparpillés un peu partout dans la salle
- **Boîtier électrique :** Des fils électriques devront être branchés au bon endroit pour déclencher un évènement.
- **L'alchimie :** Mettre en place les interactions avec la table d'alchimie présente dans la salle.

- **Puzzle** : Grâce aux énigmes résolues le joueur pourra récupérer des pièces qui lui permettront de résoudre le puzzle.
- **Interrupteurs** : Le joueur devra trouver des interrupteurs cachés à activer.
- **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 comme le PC virtuel).
- **Ranger une bibliothèque** : Le joueur sélectionnera deux livres pour les inverser et remettre la bibliothèque en ordre.
- **Jeu d'échec** : Mettre en place les interactions nécessaires au fonctionnement de l'énigme du jeu d'échec.
- **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 une carte indiquant leur emplacement.
- **Énigmes visuelles** : Des questions de type charades ou rébus seront placés dans la pièce. Le joueur devra trouver la réponse du problème et l'écrire dans un PC virtuel.

### 5.3 L'intelligence artificielle

Cette partie du projet est étroitement liée aux énigmes décrites ci-dessus. Grâce aux quelques énigmes que nous souhaitons intégrer, nous pourrons développer un premier algorithme fonctionnel. Celui-ci devrait être en capacité de choisir aléatoirement différentes énigmes, jeux et missions et les agencer dans un arbre dont les noeuds externes sont les jeux et les énigmes et les noeuds internes des missions. C'est cette arbre qui donnera l'ordre de résolution de notre salle en remontant niveaux par niveaux comme L'indique le schéma ci-dessous.



## 5.4 Finitions

### 5.4.1 Physique

La majorité des actions à faire dans la partie physique est essentiellement constitué de légers ajustements et de correction de bugs.

Comme par exemple le fait que le joueur traverse occasionnellement quelques textures dont il ne devrait pas. De plus, Quand le joueur monte sur un élément autre que le sol, celui-ci se retrouve légèrement en lévitation.

Il faudra bien évidemment rajouté des interactions supplémentaires entre le joueur et l'environnement dans le cadre de la réalisation d'énigmes.

### 5.4.2 Interface

Il reste a mettre en place toutes les options disponibles dans les paramètres accessible par le joueurs ainsi que certaines finitions au niveau de l'esthétique et du son lors des interactions.

### **5.4.3 multijoueur**

Pour la partie Multijoueur, nous avons principalement deux points sur lesquels nous devons travailler :

- **Correction des bugs** : Malgré les nombreuses corrections déjà apportées, certains bugs sont encore à corriger, notamment sur la déconnexion et reconnexion des joueurs.
- **Intégration aux nouvelles fonctionnalités** : Comme nous vous l'avons expliqué plus haut, nous allons intégrer quelques énigmes. Le multijoueur devra donc s'adapter a ces nouvelles fonctionnalités. Les joueurs devront par exemple être informé lorsque leurs énigmes respectives sont complétées.

## Conclusion

Notre projet se déroule comme nous le souhaitons. L'ensemble du groupe commence à bien appréhender le fonctionnement d'Unity, ce qui nous met dans de bonne condition pour la suite de ce projet.

De plus, nous avons la chance d'avoir une bonne ambiance en interne. Chaque personne est investie dans son travail et a respecté le temps imparti pour avancer suffisamment sur le projet.

Grâce à notre fluidité de travail, nous avons pris de l'avance sur une bonne partie des tâches que nous avions à réaliser. Celle-ci nous permettra par la suite de nous concentrer sur des points un peu plus compliqué à mettre en place comme les énigmes et l'intelligence artificielle. Grâce à cette avance, nous pourrons intégrer des fonctionnalités que nous n'aurions pas pu mettre en place sans ce gain de temps.

Notre jeu deviendra alors proche de sa version finale, c'est à dire un jeu compétitif en multijoueur.