

# Un rapport de projet - School Life

BoxOut

Juin 2023



Talvin Ackbaraly  
Titouan Cabello  
Ivan Huard  
Kaïs Hammache

# Table des matières

<b>1 Cahier des charges</b>	<b>5</b>
1.0.1 Qui sommes nous ? . . . . .	5
1.0.2 En quoi consiste ce cahier ? . . . . .	5
1.0.3 Quel est l'objectif de ce projet ? . . . . .	5
1.1 School Life - Présentation . . . . .	6
1.1.1 Origine . . . . .	6
1.1.2 Concept de base . . . . .	6
1.1.3 School Life dans le monde des jeux vidéos . . . . .	8
1.2 Aspects économiques et communication . . . . .	10
1.3 Organisation . . . . .	12
1.3.1 Répartition des tâches . . . . .	12
1.3.2 Graphisme . . . . .	12
1.3.3 Intelligence Artificielle . . . . .	12
1.3.4 Multijoueur . . . . .	12
1.4 Nos moyens pour développer ce jeu . . . . .	15
1.4.1 Nos moyens matériels . . . . .	15
1.4.2 Nos moyens intellectuels . . . . .	15
1.5 Conclusion . . . . .	16
<b>2 Ivan</b>	<b>16</b>
2.1 Première Soutenance . . . . .	16
2.1.1 Multijoueur . . . . .	16
2.1.2 Création de partie . . . . .	18
2.1.3 Rejoindre une partie . . . . .	18
2.1.4 Départ de Léo . . . . .	19
2.1.5 RéPLICATION . . . . .	19
2.1.6 Utilisation des serveurs d'UnityServices . . . . .	20
2.2 Deuxième Soutenance . . . . .	22
2.2.1 Objectif . . . . .	22
2.2.2 Résolution de bugs : . . . . .	22
2.2.3 Mise en place du Timer : . . . . .	23
2.2.4 Encore des problèmes : . . . . .	24
2.2.5 Mur invisible et ses problèmes : . . . . .	25
2.2.6 Autres bugs : . . . . .	25
2.3 Soutenance Finale . . . . .	26
2.3.1 Loading Screen : . . . . .	26
2.3.2 TP quand attrapé : . . . . .	27
2.3.3 Menu Pause : . . . . .	28
2.3.4 RéPLICATION du son : . . . . .	28
2.3.5 Système de shop : . . . . .	29
2.3.6 Headphone (ou « casque audio ») : . . . . .	29
2.3.7 Et après ? . . . . .	30
2.3.8 Modélisation de la cour : . . . . .	30
2.3.9 Récit de réalisation : . . . . .	30

<b>3 Titouan</b>	<b>31</b>
3.1 Première Soutenance . . . . .	31
3.1.1 Créer une map en 3D de A à Z . . . . .	31
3.1.2 Quel logiciel utiliser pour créer une map ? . . . . .	31
3.1.3 Créer un map selon les proportions du bâtiment C . . . . .	31
3.1.4 Création des niveaux . . . . .	33
3.1.5 Importer mon projet sur Unity . . . . .	34
3.1.6 Retour . . . . .	35
3.2 Deuxième Soutenance . . . . .	35
3.2.1 Intégration des IA sur la map . . . . .	35
3.2.2 Le mode de jeux . . . . .	36
3.2.3 Finalisation de la map du mode de jeux . . . . .	37
3.3 Soutenance Finale . . . . .	38
3.3.1 Finalisation du système de tâches et son intégration au mode de jeu . . . . .	38
3.3.2 Tâche 2 . . . . .	38
3.3.3 Tâche 3 . . . . .	38
3.3.4 Tâche 4 . . . . .	39
3.3.5 Répartition et réinitialisation des tâches . . . . .	39
<b>4 Kaïs</b>	<b>39</b>
4.1 Deuxième soutenance . . . . .	39
4.1.1 Mise en contexte . . . . .	39
4.1.2 La conception du site . . . . .	40
4.1.3 La première page . . . . .	40
4.1.4 La deuxième page . . . . .	41
4.1.5 La troisième page . . . . .	41
4.1.6 Serveur et Hébergement . . . . .	42
4.1.7 Design . . . . .	42
4.2 Soutenance Finale . . . . .	42
4.2.1 L'amélioration du site . . . . .	42
4.2.2 Player Preferences . . . . .	43
<b>5 Talvin</b>	<b>43</b>
5.1 Initiation au Projet . . . . .	43
5.2 Fonctionnement d'un Personnage . . . . .	44
5.3 Intelligence Artificielle . . . . .	45
5.3.1 Création . . . . .	45
5.3.2 Premier problème et solution . . . . .	45
5.3.3 Finalisation . . . . .	46
5.3.4 Import des IAs sur le Multijoueur . . . . .	47
5.4 Inventaire . . . . .	47
5.4.1 Référencement . . . . .	47
5.4.2 Création . . . . .	48
5.4.3 Encore des problèmes... . . . . .	48
5.4.4 Partie 2D   Visuel . . . . .	48
5.4.5 Ressenti Personnel . . . . .	49

5.5	Items . . . . .	49
5.6	Intégration des tâches au Multijoueur . . . . .	50
5.7	Batiment A . . . . .	50
5.7.1	Modélisation . . . . .	50
5.7.2	Textures . . . . .	51

# 1 Cahier des charges

## 1.0.1 Qui sommes nous ?

„ Nous sommes 3 étudiants de SUP à EPITA, qui ont formé un groupe afin de réaliser un jeu vidéo dans le cadre du projet de S2.

Le nom de notre groupe/studio de développement est BoxOut. Est notre motto est le suivant : *"Think out of the box."* Ce groupe se compose de : Ivan Huard - Talvin Ackbaraly (Léo Harlay maintenant parti du groupe) et Titouan Cabello, rassemblés par leurs idéologies et passions communes.

## 1.0.2 En quoi consiste ce cahier ?

Dans ce cahier des charges, nous allons vous faire découvrir notre projet de C#. Vous comprendrez son fonctionnement, mais aussi son origine et le déroulement de son organisation. Nous l'avons donc découpé en deux grandes parties :

- Tout d'abord nous presenterons ce qu'est le jeu en lui même, en quoi il consiste, et quels sont les aspects économique du jeu et comment nous comptons communiquer du jeu.
- Puis nous entrerons dans la partie technique du projet en expliquant comment nous avons réparti nos tâches et quels seront les outils que nous seront susceptible d'utiliser

## 1.0.3 Quel est l'objectif de ce projet ?

L'objectif de notre projet, est de pouvoir commercialiser le jeu tout en faisant prendre du plaisir aux joueur, et de rendre ce jeux accessible à de nombreuses plate-forme, afin que quiconque sur Windows puisse y jouer et prendre plaisir sur notre jeu.

Tout d'abord, comme tout développeur, notre principale source de motivation est le fait de voir le résultat final de notre projet. En effet, chaque instant où nous le développerons, cette envie en nous, de enfin pouvoir le tester s'agrandira de plus en plus. Alors oui, le principal objectif de notre projet est de le finir, peu importe les moyens utilisés et le temps donné, afin de enfin pouvoir le partager et le commercialisé.

Il nous semble tout de même important de préciser que notre jeu est à but lucratif. Certes ce projet nous demandera une quantité de temps vraisemblablement très importante, et il nous semblait donc important d'avoir une nouvelle source de motivation pour créer la meilleure version possible du jeu. Évidemment, nous avons très longuement réfléchi à cet aspect, ainsi qu'à la façon dont nous allons monétiser notre jeu.

## 1.1 School Life - Présentation

### 1.1.1 Origine

La provenance de cette idée de projet est quelque peu particulière. En effet, il y a quelques années, au collège, Léo et un ami à lui, grâce à leur passion pour les jeux de stratégies et d'infiltration, et leur imagination débordante, s'étaient imaginés un jeu, auquel ils jouaient pendant les récrés. Au lieu de faire comme tout le monde et de jouer au foot dans la cour, ceux-ci, préféraient s'imaginer être des espions, infiltrés dans leur école. Ainsi, au lieu de descendre jouer pendant la pause du matin, ils restaient au RDC, et s'amusaient à tenter de monter et traverser chaque étage sans être vu. Si jamais ils se faisaient remarquer par un professeur ou un surveillant, ils devaient redescendre d'un étage et recommencer de là, jusqu'à atteindre le dernier étage, le 3ème, afin de gagner la partie. Ainsi naquit à cette époque, dans la tête de ses deux bambins, l'idée d'en faire un jeu vidéo, beaucoup plus complet, avec un gameplay plus poussé, fun et surtout, en multijoueur.

Bien que leur esprit de créativité leur avait déjà permis de créer un jeu de cartes à collectionner et à jouer (dans le style de Pokémons ou YuGiOH), ainsi que plusieurs jeux de société, un jeu vidéo était beaucoup plus compliqué à réaliser, puisque qu'il nécessitait des connaissances techniques qu'ils n'avaient pas.

De ce fait, cette idée en resta une, mais ne fut pas oubliée, gardée bien au chaud dans un coin de la tête de Léo. Ainsi, quand pour le projet de S2 de EPITA, Léo apprend qu'il fallait créer un jeu vidéo, il se dit que c'était l'occasion, mais cette fois-ci avec plus de connaissances et de maturité, de concrétiser cette idée, qui à l'époque, le faisait tant vibrer.

Il proposa donc son idée de jeu à son groupe, qui fut retenu. Bien entendu, il a fallu l'adapter, la réinventer en lui apportant différentes fonctionnalités amusantes et intéressantes, mais surtout, un but, afin qu'elle soit plus que simplement de monter/descendre des étages sans se faire voir.

Ainsi, il s'agit désormais d'un jeu multijoueur loufoque, fun et sans prétention, qui se veut offrir des soirées de rire (ou de farm), entre amis.

### 1.1.2 Concept de base

Comme nous devons bien commencer quelque part, et puisque nous devons vous donner envie de jouer à notre jeu, laissez moi entamer ma présentation par un point que nous trouvons important, mais surtout très intéressant. School Life, est un jeu de type "School RP", qui prend place, à EPITA. Oui oui, les élèves d'EPITA pourront effectivement incarner leur propre rôle dans ce jeu ou le but est de réussir son année.

Le jeu se déroule dans l'établissement ÉPITA (modélisé à la sauce BoxOut), et le joueur incarnera donc un étudiant ayant pour but d'avoir la moyenne durant les qcm afin passer son année.

Pour cela deux moyens seront possible : suivre les cours (comme tout élève de la vraie vie ferait), ou bien..., voler le sujet du QCM qui suit, afin d'avoir la meilleure

note.

Durant la période entre deux QCMs (il y'en a un toutes les 10 minutes) les élèves devront donc, par tout les moyens, se préparer afin d'avoir une bonne note (caractérisée par un score sur 20 qui sera ensuite transformé en moyenne générale) afin de pouvoir valider leur année.

Ce qui représentera les cours sera rendu volontairement "ennuyeux" afin de pousser les joueurs à aller faire le vol de sujet qui sera considéré comme le mode de jeu principal, c'est dedans que se porte tout l'intérêt du jeu, il s'agit du cœur du gameplay, le reste permettant de donner un contexte et une raison de faire le gameplay principal. C'est surtout la partie la plus amusante et pour laquelle les joueurs joueront au jeu. Cependant School life sera également un lieu d'amusement où les joueurs pourront jouer, se parler ou rivaliser entre eux, la façon dont ils joueront pourra aussi être un moyen de sociabiliser grâce au chat(vocal de proximité) présent et accessible à tout joueur. Le gameplay deviendra donc finalement peut - être secondaire pour beaucoup de joueur.

On va pouvoir distinguer deux map principales :

Il y aura la map principale la ou tout le monde spawn au début et ou les joueurs feront leurs cours, dans cette map seront cachés des objets secrets que vous pourrez prendre et garder dans votre inventaire jusqu'à ce que vous vous déconnectiez (ou, pour les objets à usage unique, de les utiliser en partie de vol de sujet) . Les objets seront à un endroit caché et chacun pourra en prendre un, par exemple si le joueur A prend l'objet C et que le joueur B ne l'a pas encore pris alors il pourra le prendre quand même (1 objet chacun).

Il y aura la map multi qui sera à part, un point de matchmaking sera ajouté à la map principale, mais une fois la partie commencée nous spawnons dans la map multi qui elle, n'est pas accessible dans la partie "sandbox" du jeu, en dehors du gameplay de vol de sujet donc. Ici aussi on pourra prendre des objets pour combattre/faire des diversions contre ceux qui nous empêchent de voler les sujets. cependant ces objets seront perdus à la fin de la partie. Le but de ce mode de jeu étant d'offrir une expérience amusante, et décalée, les joueurs devront tenter de monter les étages du bâtiment C, afin de récupérer le sujet du prochain QCM au dernier étage. Un affrontement aura donc lieu entre les profs/faillots, et les étudiants qui s'affronteront. Vous pourrez choisir l'approche de la discréption, monter les étages sans être vu ni entendu, ou au contraire décider d'aller au "front", et vous faire courir après par le groupe adverse, tout en tentant de les piéger, avec vos objets ou "armes" type sarbacane maison, ou pied de biche trouvé dans le sous sol ( qui permet aussi d'ouvrir des portes fermées par exemple).

Fonctionnement du vol de sujet :

Le vol de sujet commence avec 2 minutes de matchmaking, si il n'y a pas assez d'ennemis (et d'ennemis uniquement) alors on ajoutera des bots pour pas donner trop facilement le sujet au joueur. Une fois le matchmaking terminé les joueurs auront 8 minutes pour voler le sujet sans se faire choper par le prof (bot)/fayot (joueur réel), il faudra qu'il y ai au moins 3 joueurs voulant voler le sujet pour qu'un autre

joueur puisse être fayot. Plus il y aura de joueur plus il y aura de fayot et moins il y aura de prof (bot).

Fonctionnement des minis-jeux :

Tâches à la Among Us, chaque tâche va représenter chaque cours. Il y aura quelque tâches par cours, on fera en sorte d'en faire 20 en tout histoire que les tâches soient variées. Ceux qui apprendront leurs cours auront du temps additionnel dans la map en attendant que le vol de sujet finisse. Pendant ce temps ils pourront explorer le bâtiment A : faire des achats d'objet avec les notes obtenues aller aux pnj illégaux (prof corrompu) vente de sujet à 5 points (sur 10) avec 1 chance sur 2 d'avoir 20 ou 0 au prochain qcm, et plein d'autre trucs cools...

### 1.1.3 School Life dans le monde des jeux vidéos

Inspirations : Garry's mod est l'un des premiers jeux dans lequel le Role-Play a une place fondamentale. En effet le mode de jeu DarkRP de Garry's mod regroupe tous les différents univers de ce mode de jeu. Le DarkRP est un mode de jeu multijoueur où le joueur incarne un personnage "de la vraie vie" dans un monde ouverte. C'est au joueur de donner une personnalité à son personnage et de rendre sa vie plus trépidante. Parmi ces univers, il existe un nommé "school RP" dont la carte est basée sur une école dans laquelle plusieurs rôles sont disponibles (différents professeurs ou différents types d'élèves). Le joueur sera donc mené à suivre une vie d'étudiant ou de professeur au sein d'un établissement scolaire accompagné de mini-jeux.



Crab Game est un jeu multijoueur publié en octobre 2021 par le youtuber dani. Grâce à l'engouement autour de la série Netflix "Squid Game" duquel il s'inspire, le jeu est propulsé en quelques temps sur toutes les pages de recommandation steam et restera l'un des jeux les plus joués de la plateforme pendant quelques mois. Le jeu s'inspire donc de l'univers de Squid Game avec des graphismes simples, géométriques et "cartoonesque". Le but, à l'image de Squid Game, être le dernier en vie parmi les 35 participants. Pour ce faire les joueurs devront participer à des mini-jeux inspirés aussi de la série. À l'issue de chacun de ces mini-jeux un certain nombre de joueurs sera éliminé jusqu'à ce qu'il n'en reste qu'un.



Yandere Simulator, un jeu où une étudiante doit accomplir des tâches insolites dans son école afin d'arriver à ses fins, notre jeu se rapproche de Yandere Simulator non pas dans le meurtre mais dans la vision "sans tabou" de création de jeu et spécialement qui a lieu dans une école



Pour ce qui est du tout premier jeu reconnu RolePLay, il s'agit de DnD. Sorti en 1975, il est développé par Gary Whisenhunt et Ray Wood. Il est le précurseur des futurs jeux de rôles, certaines de ces mécaniques de gameplay sont aujourd'hui devenues des standards du jeu vidéo de roleplay. Il connut un gros succès et fut joué jusque dans les années 1980 malgré son retard graphique.



School Life, comme son nom l'indique, est un jeu dans lequel le joueur incarne un élève dans le campus de Villejuif. Ce jeu est un multijoueur Role Play, ce qui signifie que le joueur est capable de communiquer oralement avec les autres joueurs. Afin de donner l'atmosphère la moins sérieuse possible au Role Play et au jeu en lui-même, nous avons choisi d'utiliser des graphismes simples et "cartoonesques" (low poly).

## 1.2 Aspects économiques et communication

Qu'est-ce qu'un jeu sans joueurs ? Comme vous l'aurez compris, nous allons devoir faire de la communication, afin d'attirer l'attention de potentiels joueurs afin qu'ils téléchargent et jouent à notre jeu. Alors, nous nous sommes demandé : Qu'est-ce qui fait que nous souhaitons installer et jouer à un jeu ? Nous avons rapidement trouvé x points :

- Premièrement, le nom. Ce sera le premier contact entre l'utilisateur et notre projet. Nous avons opté pour le nom " School Life " car il désigne pleinement le thème de notre jeu, sans pour autant dévoiler ses mystères, ce qui attirera inévitablement les utilisateurs.
- Deuxièmement, le trailer. Généralement présents sur les boutiques de jeux, nous montrant des gameplays plus que envieux, ou des fonctionnalités à notre jeu plus attrayantes, il est la cause de l'installation de notre jeu ou non. Ainsi, nous créerons un trailer qui dès la première seconde, fera dire au joueur que c'est le jeu dont il a besoin d'essayer.
- Troisièmement, le prix. En effet, le prix est une des qualités importantes de notre jeu, puisque même s'il est à but lucratif, il sera gratuit dans l'intégralité de ses fonctionnalités, c'est-à-dire qu'il ne sera ni sous la forme d'un " Pay To Win ",

ni d'un " Pay To Play ".

Ces trois points venant d'être cités sont effectivement important, mais il est important de savoir où ils seront disponibles. Comme vous le savez déjà, notre jeu devra être accompagné d'un site Internet, et celui-ci nous aidera à en faire la promotion. Nous ferons en sorte de tomber sur le trailer dès lors que nous arrivons sur la page, avec toutes les informations comme le titre ou bien le prix du jeu mis en avant pour directement attirer l'attention du potentiel futur joueur.

Mais ce n'est pas tout ! Vivant dans une époque où les réseaux sociaux sont au centre de notre monde, comment ne pas créer des réseaux sur toutes les principales plateformes utilisées de nos jours, à notre projet. En effet, des réseaux sociaux comme Instagram, et surtout Tiktok, peuvent faire exploser notre projet en un clin d'œil, jusqu'à faire parfois des millions de vues.

Peu importe son sujet, un projet a toujours un aspect économique important à gérer, qu'il soit à but lucratif ou non.

Certaines dépenses sont nécessaires, pour le bon déroulement de la création de notre projet. Nous envisageons premièrement de dépenser 100€ dans un service Steam afin de pouvoir commercialiser notre jeu, mais également pour pouvoir gérer toute la partie autour du multijoueur plus facilement. De plus, comme vous le savez déjà, nous aurons un site internet, ce qui implique les hébergements de celui-ci à payer. Mais pour ne pas trop étaler les dépenses, dans le jeu, nous n'hébergeront aucun serveur, et ce seront les joueurs qui devront eux-mêmes HOST les parties.

Dans notre cas, étant un projet à but lucratif, nous souhaitons obtenir profit dessus. Mais comme nous avons pu vous le préciser précédemment, notre jeu ne sera ni un " Pay To Win ", ni un " Pay To Play ". En effet, notre jeu contiendra une fonctionnalité similaire à des multitudes de jeux à la mode, comme Fortnite, puisqu'une boutique d'objet sera présente, qui permettra de pouvoir avoir des " skins " dans un premier temps, et d'autres choses qui arriveront avec le temps.

Mais vous devez vous demander : comment accèderons-nous à cette boutique d'objet ? Comme vous l'aurez compris, notre jeu se déroulera dans le campus de l'Epita de Villejuif, et la cafétéria présente dans le Bâtiment A sera en réalité modélisé en tant qu'une boutique d'objet, dans laquelle il nous faudra interagir avec un PNJ afin de pouvoir acheter un quelconque objet présent dans la boutique.

## **1.3 Organisation**

### **1.3.1 Répartition des tâches**

Bien que nous ayons tous des facilités dans certains domaine nous feront tout de même en sorte que chacun de nous passe par toutes les étapes de la création du jeu afin de ne pas laisser un membre du groupe seul avec son problème (nous utiliserons aussi la hierarchie responsable/suppléant).

### **1.3.2 Graphisme**

Dans cette section le responsable sera Léo, ayant un passé de créatif et étant passionné par le monde de la modélisation, bien qu'autant attaché à la programmation, il sera le leader de cette partie

### **1.3.3 Intelligence Artificielle**

La section Intelligence Artificielle sera principalement dirigée par Talvin et Ti-touan. Étant passionnées tout deux d'IA ils seront dans cette section au coeur de leurs passion mutuelles, ainsi durant le projet ils se répartiront (entre-autres) les tâches liées à l'intelligence artificielle.

### **1.3.4 Multijoueur**

La partie multijoueur du projet est très présente, puisqu'elle est le fondement de l'intérêt du jeu, c'est donc Ivan qui va s'occuper de mettre en place la partie réseau du jeu puisque celui-ci est déjà familier avec la création de jeu-vidéo.

Afin d'avoir une meilleure idée de ce que nous devions faire nous avons divisé notre projet en plusieurs parties afin de faire l'inventaire des tâches à réaliser. Nous avons évalué ensuite la priorité des tâches en fonction de leurs difficultés pour finalement se les répartir sur toute la durée de la construction de notre jeu.  
Nous avons alors un tableau à 2 entrées : tâche/personne afin de clarifier la situation.

	Ivan	Talvin	Titouan
<b>Vol de sujet</b>			
Système de tâches	S	R	
Système de tp en permanence		S	R
Insérer un timer InGame (fin partie)	R		S
Spawn aléatoire des objets	R		S
Spawn aléatoire du sujet	R		S
Créer la zone de TP (permanence)	S	R	
Créer les armes et leurs effets sur les joueurs	S	R	
<b>Apprentissage des cours</b>			
Insérer un timer InGame (fin cours)	R	S	
Créer les minis-jeux		S	R
Créer les armes et leurs effets sur les joueurs		R	S
<b>Graphisme</b>			
Textures (utilisation de librairies)	S		R
models 3D (Joueur, batiments, objets...)	S		R
animations (utilisation de mixamo)	R	S	
créations des UI, HUD, affichage des points. . .		S	R
<b>Réseau</b>			
Lobby général pour rejoindre des parties	R	S	
lobby propre au vol de sujet	R		S
<b>IA</b>			
IA des profs		R	S

TABLE 1 – Tableau de répartition des tâches

Afin de se projeter dans le temps un tableau à 2 entrée tâches/soutenance nous est aussi très utile afin d'optimiser notre temps de travail. Ainsi nous avons prévu de faire certaines tâches plus urgentes aux premières soutenances puis en évoluant vers la fin du temps imparti il nous restera les tâches les plus bénigne.

	Soutenance 1	Soutenance 2	Soutenance 3
<b>Vol de sujet</b>			
Système de tâches	40	80	100
Système de tp	30	70	100
Insérer un timer InGame	20	70	100
Spawn aléatoire des objets	20	70	100
Spawn aléatoire du sujet	20	60	100
Créer la zone de TP	30	70	100
Créer les armes	10	80	100
<b>Apprentissage des cours</b>			
Insérer un timer InGame	10	80	100
Créer les minis-jeux	10	70	100
Créer les armes	10	80	100
<b>Graphisme</b>			
Textures	50	70	100
Models 3D	30	60	100
Animations	30	70	100
Créations des UI	40	90	100
<b>Réseau</b>			
Lobby général	40	70	100
Lobby (vol de sujet)	40	70	100
<b>IA</b>			
IA des profs	60	80	100

TABLE 2 – Tableau des soutenances (en%)

## **1.4 Nos moyens pour développer ce jeu**

### **1.4.1 Nos moyens matériels**

Pour développer ce jeu nous utiliserons donc unity en guise de moteur de jeu. Tout d'abord car c'est un outil très puissant et très accessible pour les débutants, ensuite parce que ce langage nous permet de programmer en C# notre langage de programmation orienté objet de prédilection. Nous utiliserons aussi le logiciel Blender afin d'être libre en terme de création de modèles 3D afin d'avoir un visuel qui nous ressemble.

Dans un soucis de temps limité il y a une possibilité d'utiliser un service de steam afin de faciliter la création du multijoueur du jeu notamment pour la réPLICATION et le chat vocal de proximité.

### **1.4.2 Nos moyens intellectuels**

Nous allons aussi évidemment nous aider d'Internet afin de réaliser à bien ce projet. Nous auront recours à de nombreuses ressources tel que les documentations de C#, le forum mis en place par unity afin d'y aider ses développeurs et aussi les tutoriels qui sont à notre disposition sur youtube.

## 1.5 Conclusion

Afin d'apporter une conclusion, et un point de vue général sur l'ensemble du projet. Nous pouvons résumer celui-ci en 3 points.

- Tout d'abord, School Life est un jeu de type " School RP ", qui prend place dans les locaux de Epita situés au 66 Rue Guy Moquet, à Villejuif. Il s'agit donc d'un jeu multijoueur. Il sera accessible à un grand nombre de joueur, puisque sa direction artistique se veut simple, colorée, et rigolote. Le principe est simple, les joueurs incarnent donc leur propre rôle : " des élèves de Epita ", et doivent réussir leur année, pour cela, ils peuvent soit suivre les cours dans des salles de mini jeux, soit faire la partie principale (et la plus fun) du gameplay : le vol de sujet. A intervalle de temps régulier, des QCM ont lieu, et une note est ajoutée à la moyenne du joueur, dépendant de la réussite de ses cours et/ou du vol du sujet.

Bien que ce jeu ne soit pas un AAA, il reste pour le moins ambitieux. En effet, il est prévu qu'il comporte beaucoup de fonctionnalités (cachés ou non), mais aussi beaucoup d'objets, de modèles 3Ds, de personnages, de minis jeu, etc.

- Le planning sera donc extrêmement important, et chacun devra donner beaucoup de temps de travail, de ténacité et surtout de son cœur au projet, afin qu'il soit réalisé dans les temps. Celui-ci à été organisé dans un tableau en fonction des différentes soutenances qui auront lieu au cours du projet, et comporte tout ce qui doit être réalisé avant chaque soutenance. Vous pouvez le retrouver en page 8-9.

- Le développement se fera sur Unity en C#, et à l'aide de Blender pour la réalisation des modèles 3Ds. Ce projet nous tient particulièrement à cœur, d'une part parce qu'il est difficile et challengeant, mais d'autre part parce qu'il est le premier projet concret d'une longue série. Nous avons aimé l'imaginer, nous aimerais encore plus le programmer. Nous voudrions finir ce cahier des charge avec une citation, car comme l'a dit un jour un grand studio :



## 2 Ivan

### 2.1 Première Soutenance

#### 2.1.1 Multijoueur

Ayant déjà une expérience avec Unity, car j'avais déjà réalisé des nombreux jeux, j'ai voulu découvrir une nouvelle facette du jeu vidéo : le Multijoueur. Je ne savais

absolument pas dans quoi je me lançais, c'est d'ailleurs pour cette raison que j'ai été rapidement surpris d'avoir pris plus de temps à essayer de comprendre tout le système de multijoueur derrière un jeu plutôt que de réellement le coder.

En effet, lorsque j'ai appris qu'il fallait héberger nos joueur, de nombreux choix se sont offert à moi. Tout d'abord il fallait choisir le type d'hébergement, est-ce que nos joueurs seront hébergés sur un serveur dédié ? Pour un soucis d'économie d'argent, non. J'ai alors décidé qu'il fallait que le joueur devrait héberger lui même le serveur sur lequel il voudrait être le "host" comme je l'avais déjà sur de nombreux jeux auparavant.

C'est à ce moment la que j'ai alors commencé à me renseigner sur la manière de créer un tel service, et une fois de plus je faisais face à de multiples choix, je pouvais choisir le service proposé par de nombreuses entreprise comme ce que propose PUN2 ou Mirror mais dans un soucis de simplicité, et car il y avait beaucoup de documentation à ce sujet j'ai finalement adopté les services d'unity qui sont également relativement généreux en bande passante gratuite. J'ai commencé à me familiarisé avec ces services grace à de nombreux tutoriels comme :

- [https://www.youtube.com/watch?v=3yuBOB3VrCkab\\_channel=CodeMonkey](https://www.youtube.com/watch?v=3yuBOB3VrCkab_channel=CodeMonkey)
- [https://www.youtube.com/watch?v=3yuBOB3VrCkab\\_channel=CodeMonkey](https://www.youtube.com/watch?v=3yuBOB3VrCkab_channel=CodeMonkey)
- [https://www.youtube.com/watch?v=3yuBOB3VrCkab\\_channel=CodeMonkey](https://www.youtube.com/watch?v=3yuBOB3VrCkab_channel=CodeMonkey)
- <https://docs-multiplayer.unity3d.com/>

Comme on peut le constater le YouTuber CodeMonkey et la documentation très riche que nous propose unity vis à vis de ses services m'ont été d'une grande aide durant l'apprentissage et la réalisation de la partie multijoueur de notre projet.

Pour ce faire j'ai utilisé deux services que unity nous propose, le Relay et le Lobby. En effet grâce à ces deux services j'ai pu connecter les joueurs entre-eux à l'aide des lobby puis les faire se rencontrer et interagir entre-eux grâce aux relay. Le relay attend alors l'intervention d'un joueur pour héberger une partie puis se met finalement à ouvrir une connexion par le biais du joueur "hébergeur".

Unity nous propose en plus un système de Lobby et Relay à code, afin de pouvoir rejoindre ses amis dans une partie en possession du code, j'ai alors fait en sorte que chaque joueur entrant dans une partie ai à disposition son code de partie pour pouvoir jouer avec des amis.

La grande difficulté de la réalisation de la création du Relay et des Lobby (services d'unity utilisés) ont été la résolution de bugs. En effet en utilisant des outils qui m'était nouveaux la moindre erreur l'était également ce qui rajoutait de longues heures de recherche. J'ai vite remarqué que ce service que propose Unity est récent car de nombreux bugs que j'avais avait déjà été relevé il y a peu de temps par d'autres utilisateurs et pas encore patchés.



Les moments où ces erreurs apparaissaient étaient intéressants car ces erreurs m'apprenaient à faire autrement mais c'était en même temps assez inquiétant car je ne savais pas combien de temps j'allais devoir consacrer pour y remédier et comme chacun sait, à Epita, chaque seconde de temps libre est précieuse.

### 2.1.2 Crédit de partie

Un choix à également faire lors de la création de partie et de la forme que prendrait le lobby.

J'ai alors commencé par visiter différentes approches, et ai essayé d'intégrer la création de serveur sans l'utilisation de lobby mais je me suis vite rendu compte que sans lobby il était compliqué de joindre le serveur sans code. Devoir avoir un code pour jouer au jeu est très, et même trop, privatif, c'est pour cette raison que j'ai intégré une manière simple et efficace de joindre un serveur, sans code, et juste avec l'intervention d'un clic sur un bouton « Rejoindre une partie ».

Ainsi lors de la création d'une partie il est créé en arrière plan un lobby, sans que le joueur le sache, qui va pouvoir stocker le code de la partie (car chaque lobby peut stocker des données qui lui sont propres).

### 2.1.3 Rejoindre une partie

- Ainsi pour rejoindre une partie, on laisse au joueur deux options :
- soit il rejoint une partie grâce au code d'un de ses amis
  - soit il clique sur « Rejoindre une partie » et il est automatiquement relié à une partie déjà existante (ou en crée une si il n'y a aucune partie créée)

Le code caché derrière le bouton « Rejoindre une partie » est relativement simple, il utilise la fonction `quickJoinLobby()`, créée par Unity, et quand le lobby est rejoint le code va chercher dans les données du lobby pour récupérer le code du serveur lié

au lobby rejoindra puis va simplement rejoindre le serveur à l'aide du code. Les lobbies vont donc être un intermédiaire, joueur – Relay pour les nouveaux arrivant sur le jeu School Life™.

Ce système d'intégration rapide des nouveaux joueurs au sein du jeu va rajouter plus de dynamisme car il suffira de cliquer sur un bouton pour jouer.

Il rameutera plus de joueur car grâce à ce bouton même les joueurs expérimentés pourront jouer sans l'intervention de tiers pour créer la partie et jouer avec lui.

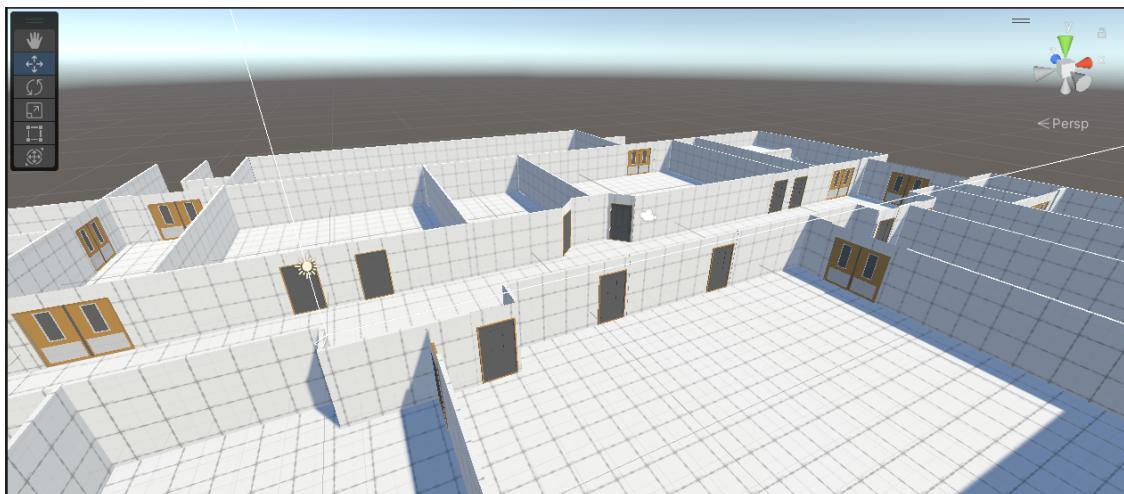
#### 2.1.4 Départ de Léo

Suite au départ récent de notre camarade de l'établissement, nous devons nous mobiliser pour anticiper ses tâches de sorte à prendre le moins de retard possible. Pour cela nous devons agir vite et efficacement, c'est pour cette raison que dès maintenant j'ai libéré le maximum de temps et ai commencé la map représentant le sous-sol du bâtiment A.

En faisant ce sous-sol j'ai surtout appris à modélisé en utilisant pro-builder. De cette manière je serais plus à l'aise au cas où à d'autre moment il faudrait d'urgence avoir une modélisation.

Selon moi l'adaptation rapide est une qualité cruciale dans un groupe, quelque soit la situation il faut pouvoir s'adapter et tirer parti de chaque situation.

Titouan a également pris part dans la réalisation des tâches de Léo, de cette manière on arrivera peut-être à ne pas prendre trop de retard sur notre avancée du projet.



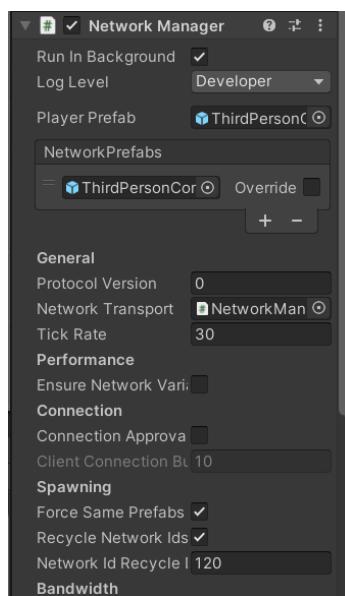
#### 2.1.5 RéPLICATION

Une des difficultés de la partie multijoueur a été la réPLICATION. En effet, pour que les joueurs aient tous la même vision d'un objet, c'est-à-dire qu'ils le voient tous apparaître et disparaître au même moment on doit stocker l'information de l'objet dans le serveur. Ensuite, en fonction des interactions que les joueurs ont pu avoir avec l'objet, le serveur va modifier l'information de l'objet puis va renvoyer l'information actualisée à tout les joueurs pour que ceux-ci soient synchronisés sur

la même "version" de la partie.

Ainsi, tout au long du projet il faudra que, pour chaque objet, le serveur stocke les informations relative à cet objet puis les renvoient au client. C'est donc ici toute la difficulté de cette partie, mettre à jour les objets, les bons et au bon moment. Heureusement, unity va également nous faciliter la tâche en prévoyant un script qui va gérer ce type d'objet "Network Manager". Celui-ci stocke une "liste" des objet à répliquer dans l'on va pouvoir remplir au fur et à mesure que des objets sont ajoutés. Grâce à ce script toute la partie "serveur" est épargnée et il nous reste plus qu'à rajouter un script propre à l'objet pour lui dire de s'actualiser sur le serveur.

Ce script "Network Manager" va avoir une place très importante dans le jeu, il contiendra le joueur principal et le fera spawn. C'est en lisant d'ailleurs ce script que j'ai compris pourquoi mon joueur n'apparaissait pas aux coordonnées (0,0,0), en effet pour une raison mystique ma prefab de joueur n'était pas intancié à (0,0,0). En regardant dans le script j'ai vu que on prenait la valeur de position de la prefab pour valeur de spawn du joueur, j'ai alors simplement changé la valeur du spawn en changeant la position de ma prefab du joueur.



Ci dessus, une image du script "Network Manager" vu depuis unity. On y voit également beaucoup d'autre paramètre ce qui en dit long sur le nombre de choses différentes que nous apporte ce script et comment il nous permet de le modifier.

### 2.1.6 Utilisation des serveurs d'UnityServices

Malgré sa grande abordabilité/son aspect relativement intuitif, UnityServices (la bibliothèque d'importation des services d'unity avec laquelle je fais la majeure partie de mon travail) contient tout des mêmes certains aspects qui nous rappelle le fonctionnement d'un serveur.

Par exemple les erreurs de débutant auxquelles j'ai pu faire face ont été de ne pas me rappeler d'allumer le serveur avant de l'utiliser. En effet pour pouvoir utiliser chaque services relatifs à unity on doit executer "await UnityServices.InitializeAsync();" avant afin que celle-ci puisse fonctionner. De la même manière le concept de "maintenir en vie un serveur" en le pingant n'a pas été très intuitif et à réussi à me bloquer pendant de longues heures. Cependant, une fois le concept compris il semble normal de ping, par exemple le lobby, pour lui dire de rester en vie.

En réalité lorsque l'on veut rejoindre un serveur à travers un relay, on récupère les données du relay du serveur sur lequel on veut se connecter et on les injecte dans "notre" propre relay. On voit bien à travers ce script comment fonctionne ce système. On voit tout d'abord comment on "se connecte" au relay à l'aide du code, ensuite on voit de quelle manière on l'injecte avec toutes les infos du relay que l'on a récupéré (IpV4,Port,ConnectionData, etc).

```
public async void JoinRelay(string joinCode)
{
    try {
        JoinAllocation joinAllocation = await RelayService.Instance.JoinAllocationAsync(joinCode);
        textOfServ.text = joinCode;
        NetworkManager.Singleton.GetComponent<UnityTransport>().SetClientRelayData(
            joinAllocation.RelayServer.IpV4,
            (ushort)joinAllocation.RelayServer.Port,
            joinAllocation.AllocationIdBytes,
            joinAllocation.Key,
            joinAllocation.ConnectionData,
            joinAllocation.HostConnectionData
        );

        NetworkManager.Singleton.StartClient();
    }catch (RelayServiceException error){
        Debug.Log(error);
    }
}
```

On peut aussi constater que j'ai dû faire appel au mot-clé "async". Il m'a été très utile car lorsque j'assigne ma variable "joinAllocation" j'attends la réponse de la fonction et le temps mis à répondre n'est pas négligeable, c'est pour cette raison que l'on utilise le mot clé "await" qui signifie "en gros" : attend que ma fonction ait répondu puis assigne la à la variable correspondante. On utilise également les mots clés "try... catch ..." au cas où une erreur survient au moment de récupérer les données du relay, si le code correspond pas par exemple, ou si les serveurs ne répondent plus. Dans ce cas je me contente d'afficher l'erreur en console mais dans la fonction quickJoin si aucun serveur n'est trouvé alors on passera dans la partie "catch" du code et alors un nouveau serveur sera créé.

## 2.2 Deuxième Soutenance

### 2.2.1 Objectif

Ma principale activité durant cet entre-deux entre la soutenance 1 et 2 aura été de faire en sorte de joindre tout nos travaux individuels sur un même projet pour éviter les incompatibilité et de surcroit ajouter (tant bien que mal) le multijoueur pour synchroniser les scènes et faire en sorte que tout les joueurs aient le même résultat.

### 2.2.2 Résolution de bugs :

Tout d'abord lorsque j'ai joint nos projet j'ai eu un premier problème. Nous avions tout créer un joueur sur le même asset (de l'asset store d'Unity) de base, cependant nous avions tous fait quelques petites modifications afin de pouvoir correctement l'utiliser dans notre projet. Lors de la jointure des projets, évidemment, unity à commencé à me sortir un farandole d'erreur toutes aussi incompréhensible, les unes des autres. Heureusement nous avions tout nos scripts fonctionnels individuellement, alors j'ai fait un choix et ai gardé le joueur de tel partie du projet et le UI de l'autre afin d'avoir une version « stable ».

Une fois encore, j'étais très loin de l'optimisation que l'on vous présente pendant l'oral de soutenance. En effet, lorsque j'en ai eu fini avec le choix il a fallu réintégrer tout ce que j'avais supprimé en faisant attention à ce que tout soit compatible avec le multijoueur (entre-autres).

J'ai du par exemple commencer à résoudre un bug avec l'animation du joueur, et ce n'était même pas encore lié au multijoueur mais juste à un problème de sol avec des tags différents. Une fois comprise l'erreur semble évidente et simple à régler, mais encore faut il comprendre l'erreur et trouver d'où elle peut bien venir.

Les problèmes de sols mal « tagué », lors du développement d'un jeu multijoueur, est le cadet de nos soucis en terme de problème d'animation. Lorsque l'on créé un jeu en multijoueur on doit envoyer en permanence son état au serveur pour que celui-ci répande cet état aux autres joueurs. Ainsi pour avoir des animations qui fonctionnent correctement on doit bien l'envoyer au serveur pour que celui le répercute à tout les autres joueurs. Encore une fois, lorsque l'on à compris comment ce système fonctionne tout a l'air simple, mais l'intégration en c-sharp l'es bien moins. Heureusement pour moi j'ai déjà fait face à ce problème avant la première soutenance et ai juste dû le réimplémenter sur le joueur puisque j'avais pris celui de Talvin qui ne s'occupait pas encore du multijoueur.

Un autre système fondamental en multijoueur est la synchronisation des objets. Si un joueur prend un objet, il paraît normal qu'un autre ne puisse plus le récupérer,

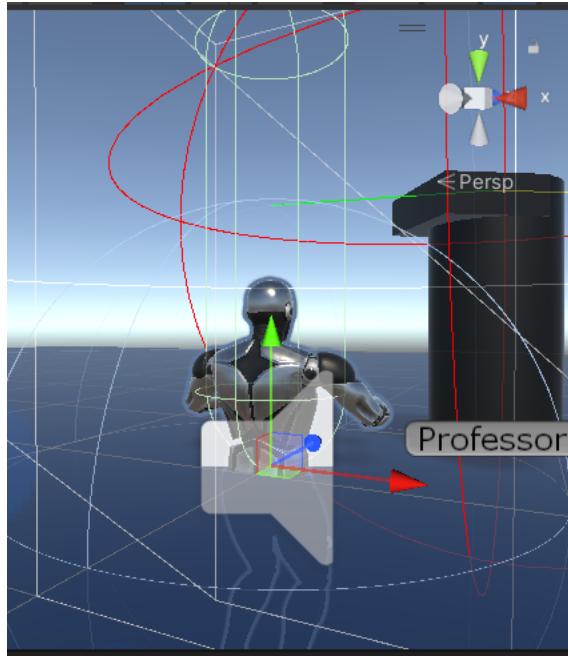


FIGURE 1 – Exemple de bug d’animation après intégration.

une fois encore, faut-il l’implémenter. Car si il y a une chose que j’ai apprise c’est que le multijoueur est loin d’être natif (en tout cas sur unity) et qu’il faut le construire de A à Z pour le moindre détail, il faudra envoyer au serveur, pour tout objets, les informations dont les joueurs auront besoin.

### 2.2.3 Mise en place du Timer :

Pour faire en sorte que les événements se passent correctement, nous avons besoin de les rythmer, c'est pour cette raison que nous avons du nous résoudre à intégrer un timer. Une fois de plus, là où un timer classique en jeu non multijoueur peut prendre 5-10 minutes à être construit, rien que pour un timer on peut au moins rajouter une heure ou deux au temps de réalisation.

Car effectivement, une fois encore, le timer devra être synchronisé avec tout les autres joueur pour que les évènements puissent l'être à leurs tour et qu'un joueur d'un côté ne passe pas son qcm pendant que l'autre vole le sujet sur le même serveur en même temps.

Ici nous avons alors utilisé la « technologie Client/Server RPC », qui nous autorise à effectuer des tâches (à créer des fonctions en clair) qui ne seront exécutée que par notre serveur (serverRpc) ou bien que par notre client (clientRpc). Outil très utile pour la réPLICATION de notre timer.

Il nous a donc « simplement » suffit de faire varier le timer par le serveur (avec un server Rpc) en décrémentant la différence de temps entre chaque appels. Puis une

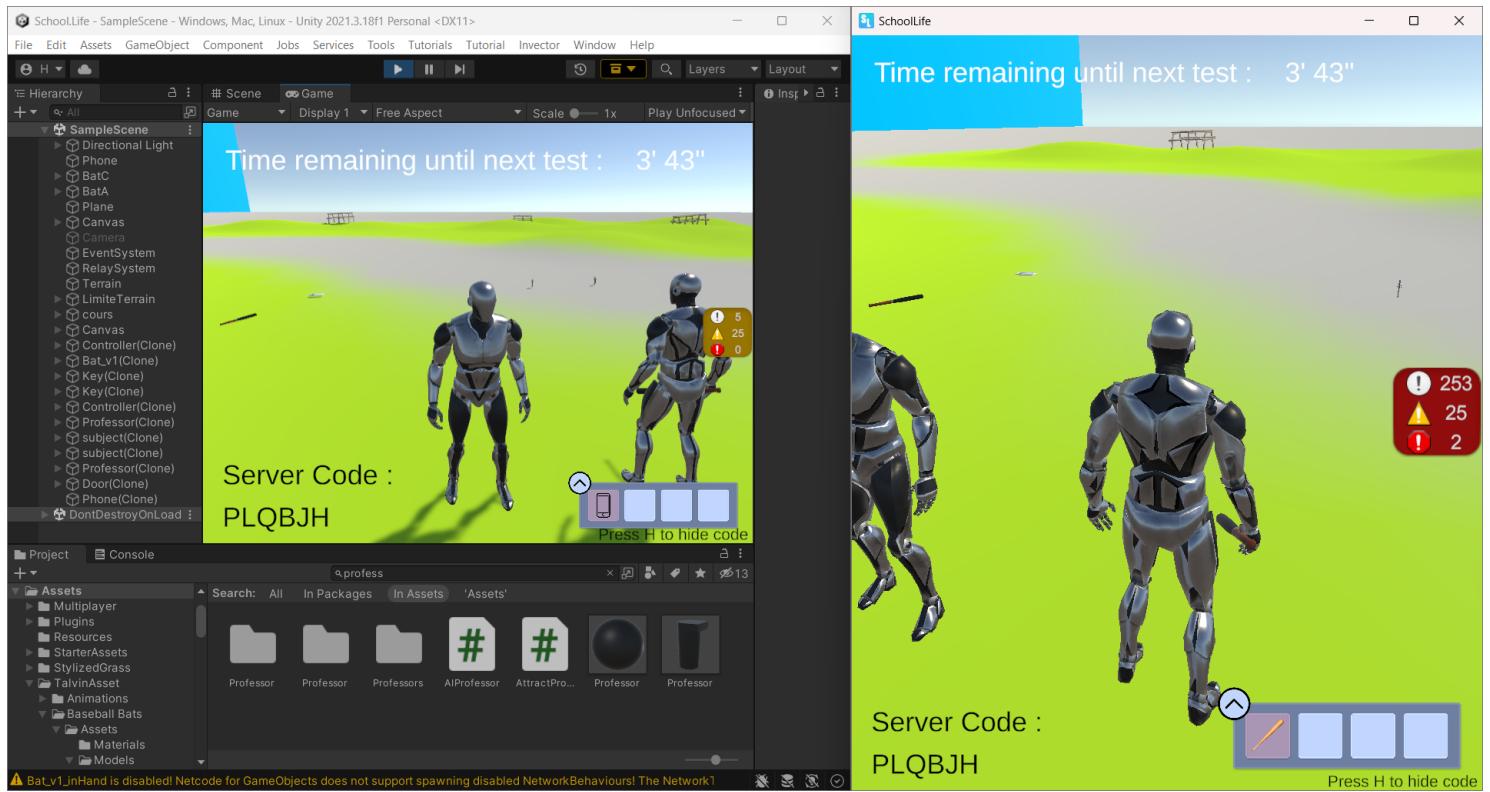


FIGURE 2 – Synchronisation du timer et des objets

fois le timer fonctionnel sur le serveur, il a fallu l'envoyer à tout les clients (dont le Host de la partie considéré comme un serveur ET un client) pour que ceux-ci mettent à jour l'affichage du timer de leurs point de vu.

Ainsi chaque joueur était synchronisé sur un timer commun, même ceux qui arrivent en cours de jeu peuvent alors voir le timer global sans erreurs. Ce timer nous servira de base pour pas mal de choses notamment le spawn des différents objets comme les sujets (une fois le vol de sujet terminer il faudra ‘refill’ le bâtiments de sujets pour le prochain qcm). C'est pour cette raison, entre-autres, que nous avons inclus une pause, laisser le temps, au jeu de faire spawn les objets de nouveau, et aux joueur de rejoindre la partie et choisir entre cours et vol.

#### 2.2.4 Encore des problèmes :

Au fur et à mesure que notre jeu prenait de l'ampleur de nouveaux bugs faisait leurs apparition.

Un qui m'a particulièrement pris pas mal de temps et sur lequel Talvin m'a aidé était l'affichage d'une boite de dialogue qui d'affiche quand on regarde un objet. Au début la boite ne se supprimait pas lorsque l'on récupérait l'objet qui lui se détruisait. Une fois ce problème réglé, un autre fit son apparition, la même boite

de dialogue pour montrer l'état de la porte ne s'affichait plus. Cette fois-ci ce fût Talvin, qui, ayant fait lui même ce système, eût réussi à résoudre ce problème.

### 2.2.5 Mur invisible et ses problèmes :

Après cela j'ai remarqué qu'il y avait une erreur non pas de code mais de game-play.

En effet, dans le vol de sujet les joueurs débloquent petit-à-petit des étages. Étant un jeu collaboratif lorsqu'un joueur parvient enfin à débloquer l'étage, celui-ci se débloque pour tout les joueurs présents dans le vol de sujet.

Ici le risque est que tout les joueurs attendent que les niveaux soient débloquer pour aller voler les sujets au dernier moment. Pour palier à ce problème nous bloquons alors l'accès au vol de sujet (présent dans le bat C, donc nous bloquons l'accès à celui-ci) pour pouvoir engager les joueurs à ne pas attendre le dernier moment au risque de ne rien avoir.

Je m'en suis alors occupé en récupérant les données du timer et en déclarant le « mur invisible » comme un network object, c'est à dire que quand il est actif, il l'est pour tout les joueurs du réseau.

Lorsque les données du timer affichent qu'il reste 3 minutes alors le bâtiment se ferme en bloquant l'accès avec le mur invisible. Je pense également à un système que nous intégrerons peut-être pour la soutenance 3, qui consiste à obliger les joueurs à faire au moins 1 tâche pour débloquer un étage afin d'avoir un sujet.

Une fois n'est pas coutume, en intégrant ce mur j'ai vu que lorsqu'un joueur A prend un objet sur un serveur et que un autre joueur B se connecte au même serveur, celui-ci voit toujours l'objet supposé détruit. Je détruisais l'objet vis à vis du serveur au lieu de lui annoncé que cet objet devais disparaître du point de vu de tout les clients, sur les clients connecté cela fonctionnait mais sur ceux qui rejoignait la partie en cours de route, ils subissaient une désynchronisation vis-à-vis du serveur. Ainsi je les ai fait despawn sur chaque client et sur le serveur, et chaque client voulant rejoindre pour chaque items devront vérifier si l'objet est toujours présent sur le serveur, si oui on ne fait rien, si non on le fait disparaître et notre jeu est désormais synchronisé comme jamais.

### 2.2.6 Autres bugs :

De nombreux bugs ont été résolu en groupe avec Talvin, je ne m'étendrai pas sur ceux-ci étant pas des bugs résolu personnellement. Cependant suite à cette session de résolution de bugs et d'intégration de certaines parties de Talvin (c'est à dire intégration de la batte et du sujet). Sur notre version commune nous avions

un problème avec l’interaction de l’inventaire qui marchais totalement sur la version temporaire, nous avons simplement opté pour reprendre ce que nous avions fait dans la version temporaire et l’inventaire marchait de nouveau. Nous nous sommes rendu compte que parfois, il suffit d’un rien pour créer une anomalie informatique.

Comme simple exemple, on peut prendre le spawn d’item. Comme nous faisions spawn nos objet de trop haut et que ceux-ci avaient une petite box collider (comme par exemple les clés ou le téléphone) ils traversaient simplement la map. J’ai bien passé une heure ou deux à chercher dans le code pour voir si je faisais mal apparaître mes objets pour finalement simplement les faire apparaître de moins haut pour avoir moins d’élan dans leurs chute (et en agrandissant aussi un peu leurs box collider afin de faciliter l’opération).

J’ai aussi du faire face à une sombre histoire de porte qu’on avait réglé avec Talvin en majorité mais il restait un bug de collision incongru. On pouvait traverser cette porte lorsqu’on était l’hébergeur de la partie. Ce qui, pour nous, n’avait aucun sens. Pourquoi diable le joueur host pouvait traverser cette porte alors que les autres client ne le pouvait pas ??

Cette résolution s’est résolue à simplement ajouter un box collider au parent de la porte (le model et la boite de collision étant en sous partie : poignée, porte, charnière, ...) Ce box collider ajouté, plus de bug de collision.

Pendant cette deuxième soutenance, j’ai aussi appris à me méfier des assets qu’on pouvait trouver sur l’asset store. Comme exemple, notre batte de baseball qui vient de l’asset store avait un problème d’affichage à cause d’un composant, le « LOD » pour LevelOfDetail, afin d’optimiser un jeu cette technologie est utilisée, ainsi, plus on est loin de l’objet plus celui-ci à une texture de mauvaise qualité pour ne pas forcer l’utilisateur à charger de trop grosse image pour rien.

Ce LOD, donc, était mal réglé, ou en tout cas, pas adapté à notre projet puisque l’objet disparaissait quand on était à un mètre de celui-ci, une fois une grand investigation inutile dans mon code je me suis rendu compte qu’il fallait simplement régler ce paramètre correctement afin de l’adapter à notre projet.

## 2.3 Soutenance Finale

### 2.3.1 Loading Screen :

Avant de me lancer dans toute programmation, il fallait d’abord avoir une interface de jeu propre afin d’avoir un plan de travail correct et continuer à travailler efficacement. Il s’en est donc suivi de nombreuse création tel que le loading screen

qui nous ont permis d'avoir un jeu propre et non pas une caméra qui se bloque pendant que l'on rejoint un serveur. Ainsi en placement simplement un canvas derrière le menu et en le détruisant à l'entrée du joueur dans la partie, il était tout de suite beaucoup plus agréable à jouer. Nous avons aussi ajouté d'autre améliorations visuelles mais j'y reviendrais plus tard dans le rapport, car comme dirais l'autre, ceci est une autre histoire.

De la même manière que dans la plupart des jeux nous avons rajouté des petites phrases, plus ou moins utile, afin de faire patienter le joueur quelques seconde de plus si le chargement pourrait s'avérer long ce qui évidemment ne pourrait être dû qui a une mauvaise connexion du client. En effet Talvin et moi avons tout fait pour que le jeu soit le plus optimisé possible et que ce genre de problème ne soit pas du à une mauvaise optimisation de jeu.

### 2.3.2 TP quand attrapé :

Pour que les professeurs aient un réel impact sur les joueurs nous n'avons pas seulement appliquer un effet visuel sur l'écran du joueur où le faire ralentir, nous avons voulu combiner les 2, d'une part et déporter le joueur en bas du bâtiment et d'autre part appliquer un effet visuel sur l'écran afin que le joueur se rende compte qu'il s'est fait attraper par le professeur.

Ce message visuel fait également comprendre au joueur que son action est considérée comme mauvaise par le jeu afin de lui faire ressentir des émotions similaires à celles qu'il pourrait ressentir dans la vraie vie s'il volait vraiment le sujet.

Pour ce faire nous avons ainsi repris le système d'Intelligence Artificielle que Talvin avait fait, cependant nous y avons apporté quelques modifications tel que la vitesse ou la vision augmenté du professeur, de quoi rendre la vie du joueur (élève) plus dure. D'autres modifications du professeur ont été effectuées comme le modèle (volontairement robotique) et l'animation qui nous semblait obligatoire avec un modèle humanoïde. Une fois le professeur calibré il nous fallait donc ensuite lui donner du pouvoir. Je me suis alors chargé de donner ce pouvoir au professeur.

Pour cela il fallait que je me plonge dans le code de Talvin afin de comprendre complètement comment celui fonctionnait pour pouvoir interagir avec et ajouter ces éléments indispensables au jeu.

Une fois que j'ai compris comment fonctionnait son système d'intelligence artificielle, je savais quand le joueur se faisait détecter, puis attraper par l'IA. Ainsi il me suffisait de récupérer les données du joueur attrapé, dans mon cas, à travers un raycast qui enregistre les données du joueur attrapé. Il me suffisait ensuite de simplement activer le bon canvas et de changer sa position vers le bas du bâtiment.

Évidemment, comme tout au long du projet tout ceci ne se fit pas sans résolution de bugs, par exemple pour une raison qui nous reste relativement obscure nous avons du désactiver le composant « Character Controller » du joueur afin qu'il puisse être téléporté. Nous avons ensuite réactivé ce composant afin, que, une fois téléporté, le joueur puisse continuer à jouer. Enfin pour ce qui est du canvas de jeu qui s'active lorsque le joueur se fait attraper, on le fait simplement disparaître quand le joueur sera prêt à continuer à jouer, une fois que celui-ci l'aura décidé, il pourra appuyer

sur n'importe quelle touche et le jeu pourra recommencer.

### 2.3.3 Menu Pause :

Comme je le disait précédemment, afin d'avoir un jeu plus agréable visuellement nous avons rajouté quelques features de jeu, en suite de l'écran de chargement se trouve donc le menu pause. Comme pour l'ensemble du jeu, nous avons voulu garder un style simple et intuitif, le menu pause est donc fonctionnel et se résume à un bouton « Resume » et un bouton « Quit » simplement pour rejoindre la partie ou la quitter (respectivement). Cependant nous trouvions tout de même ce menu un peu vide, alors nous en avons profité pour améliorer l'expérience utilisateur.

Dans nos précédentes soutenances je faisais afficher le code de la partie (celui que vous prêtez à vos amis pour qu'ils puissent vous rejoindre) directement « brut de décoffrage » dans la partie, afin d'avoir un meilleur gameplay nous avons donc décidé de mettre le code dans le menu pause avec un joli texte et une police agréable. Ainsi le menu n'est plus aussi vide qu'avant et voit son utilité se renforcer.

La petite difficulté était de gérer la visibilité du curseur, en réalité nous avons utilisé le « lock cursor » qui place le curseur au centre de l'écran et le rend invisible, cela rend l'expérience de jeu bien plus agréable. Ainsi nous avons du le désactivé/activé aux moments opportuns. Talvin approfondira dans sa partie car c'est lui qui s'est occupé de cette activation du curseur.

### 2.3.4 RéPLICATION DU SON :

Le son était un ajout auquel nous avons pensé à la dernière soutenance, notre jeu étant multijoueur nous ne pouvions évidemment pas nous contenter de simplement ajouter le son en local et seulement le joueur concerné pourrait l'entendre. C'est pour cette raison que nous avons développé le son en multijoueur. Pour cela nous avons donc utilisé un objet vide qui représente le son, à celui-ci nous avons ajouté un audio source évidemment mais aussi un network object, un composant qui va permettre de répliquer un objet à travers le réseau. Nous avons également mis un network transform qui, lui, réplique la position de l'objet. Ainsi, avec tous les outils en mains, il nous suffisait maintenant de faire apparaître l'objet au bon endroit et de faire jouer le son dès que l'objet allait apparaître.

Nous avons par exemple utilisé ce son pour l'interaction avec la batte de base ball car il nous semblait cohérent que le joueur qui se faisait taper et les joueurs aux alentours puissent aussi entendre le coup de batte. Cela permet également de pouvoir donner une ambiance au jeu.

Pendant qu'ils marchent les professeurs font aussi du bruit que tout le monde peut entendre avec leurs bruits de pas, de quoi ajouter de la pression et du suspense pour ceux qui tentent de voler le sujet.

### **2.3.5 Système de shop :**

Afin de donner une meilleure accessibilité des items aux joueurs j'ai créé un système de shop que nous avons ensuite intégré au bâtiment A à travers un PNJ. Pour la détection du joueur qui veux ouvrir le shop avec le bot j'ai simplement récupéré le même système de détection que pour les items ou encore celle des tâches. Une fois le joueur relié au pnj, pour faire apparaître le shop j'ai simplement activé un canvas présent au préalable dans le joueur. Cela de même pour les tâches, cela signifie que tout les canvas qui s'affiche dans le jeu sont présent mais simplement inactif.

Ce canvas du shop, comme la plupart de notre jeu, est aussi assez simpliste, il y contient une inscription « Shop », et des boutons avec une icône par objet et son prix. Dès que le joueur clique sur une icône celle-ci lui donne l'item en question si et seulement s'il a assez d'argent pour se l'acheter, on doit donc aussi accéder à la variable du joueur qui correspond à son argent. La difficulté ici est que notre jeu est multijoueur, tout d'abord chaque joueur doit avoir une variable distincte, puis l'accès devra se faire sur le bon joueur.

Une fois le bon joueur sélectionné, on doit lui donner l'item correspondant en fonction du bouton sur lequel il aura cliqué. Le script du shop est donc dans l'obligation de communiquer entre le bouton et la variable de prix, sans cela le script ne pourrait pas marcher.

Système de points/argent : Le système « points/argent » (car les deux sont liés) est relativement simple à comprendre. Les points vont simplement déterminer combien d'argent nous allons gagner selon la formule ( $\text{note}/20$ )\*somme maximum. Ainsi le joueur remportera la somme maximum si jamais il arrive à avoir 20/20 et proportionnellement en fonction de sa note. Ainsi le joueur aura 4 minutes pour accumuler le plus de points possible, puis, lors du qcm son gain sera calculé et son argent attribué.

### **2.3.6 Headphone (ou « casque audio ») :**

J'ai intégré un headphone afin de pouvoir faire écouter de la musique au joueur et de lui permettre de se mettre dans l'ambiance de son choix. L'intégration de ce casque audio permet un véritable changement d'ambiance et peut permettre au joueur de se motiver à try hard le jeu. Pour l'intégration de cet headphone j'ai tout d'abord voulu simplement mettre une playlist d'une heure de synthwave (libre de droit) en un seul morceau et le faire jouer. Cependant j'ai dû faire face à 2 problème, le premier, si le joueur n'aime pas la musique du début de la playlist il devra sans cesse attendre que celle-ci finisse pour écouter une autre musique, ce qui est vite dérangeant. Le deuxième problème est que unity accepte difficilement les gros fichiers audios.

J'ai alors stocké plusieurs morceaux en ligne puis ai fait une requête web vers le site d'hébergement. Ainsi, non seulement le joueur, s'il recommence le morceau, ne tombera pas forcément face au même grâce au Random.Range qui donne (quasi) toujours un nombre différent mais en plus le projet ne sera pas alourdi par d'autre

fichier audio.

### 2.3.7 Et après ?

Une fois le Timer terminé, le qcm se déclenche et de nombreuses actions prennent place. On va donc devoir interagir avec ce timer pour détecter quand déclencher le qcm. Une fois le timer récupérer on va devoir déclencher de nombreuses choses.

Tout d'abord on va vérifier si le joueur courant a le sujet dans son inventaire, si c'est le cas on place son nombre de point à 20, ayant le sujet dans les mains il sera apte à connaître toutes les réponses à l'avance. Ensuite tout les joueurs restant dans le bâtiments C, celui ou on vole les sujets, sont déplacés vers le point de spawn afin de ne pas leurs donner l'avantage sur le prochain qcm. Nous profitons de la minute de pause que l'on a pour reset l'emplacement des sujets, non récupérés et en refaire spawn le nombre qu'il faut afin d'avoir un jeu propre et qu'il reste des sujet au prochain cycle de vol de sujet.

Pour les joueurs qui ont fait les cours leurs somme sera simplement calculée en fonction du nombre de points qu'ils ont pu accumuler aux cours de leurs leçons. L'avantage de faire les cours est que cela prend moins de temps aux joueurs, ils auront donc plus le loisir de pouvoir par le suite s'amuser dans la cours avec les items ou autre.

### 2.3.8 Modélisation de la cour :

Afin de faire ma part de modélisation et de ne pas laisser mes camarades faire tout le sale boulot je me suis attelé à la modélisation du terrain, de faire une représentation réaliste de la cour, en incluant les zones d'herbes exactes, les arbres, l'herbe, etc...

Cette zone sera l'Épicentre de notre jeu, il est au centre des deux modes de jeu principaux et il correspond à l'endroit de spawn, il est donc primordial de donner une bonne image du jeu dès que le jeu arrive sur celui-ci et que la cours soit un endroit agréable à vivre, ensoleillée et facile d'accès.

### 2.3.9 Récit de réalisation :

Dans mon rapport je répète beaucoup une phrase du type « mais Talvin vous en dira plus », non pas que je délègue trop mes tâches à Talvin mais durant cette soutenance Finale avec Talvin nous nous sommes beaucoup entre-aidés que ce soit dans la résolution de bug ou de fonctionnalité annexe à part entière.

Nous avons également eu quelques désaccords avec Talvin car celui-ci voulait que le projet soit « parfait » jusqu'à la moindre petite convention, bien que je lui expliquai

qu'il n'y avait pas de bonne ou mauvaise convention de nommage. Pour ne pas nous faire perdre plus de retard j'ai essayé d'utiliser ses conventions.

## 3 Titouan

### 3.1 Première Soutenance

#### 3.1.1 Crer une map en 3D de A  Z

Au dbut du projet, lorsque les tches ont t attribues  tout le monde, je me suis propos pour crer la map car j'avais quelques ides en tte sur le parcours que devrait effectuer le joueur. Cependant,  ce moment-l, je n'avais aucune ide dans quoi je mettais les pieds. Pour moi, la cration d'une map telle que je l'imaginais me paraissait plutt simple. Mon objectif tait dsormais de crer une map qui ressemble partiellement au btiment C, mais qui serait agence de manire  rpondre aux attentes du mode de jeu. 5 tages reprsentant chacun un niveau avec des tches  effectuer pour passer au suivant.

#### 3.1.2 Quel logiciel utiliser pour crer une map ?

Je me suis donc mis  regarder plusieurs vidos expliquant quel logiciel est le plus utile pour crer une map de type low poly. Mais je n'ai trouv aucune rponse exacte, je ne savais pas exactement quoi utiliser entre Unreal Engine, Blender ou ProBuilder de Unity.

J'ai donc naturellement commenc avec Unity afin d'viter d'importer/exporter mes projets et de gagner du temps. Cependant, j'ai rencontr un premier problme. Il s'avre que c'est extrmement compliqu de crer ses propres formes gomtriques avec ProBuilder. Aprs les conseils de certains camarades, je me suis donc tourn vers Blender qui tait apparemment plus efficace dans ce domaine, mme s'il est plus compliqu  prendre en main.

Aprs quelques heures  faire le tour des vidos d'initiation  Blender sur Youtube, je me considre prt  commencer  crer notre monde. Ici, les vidos qui m'ont t le plus utiles afin de construire mon monde :

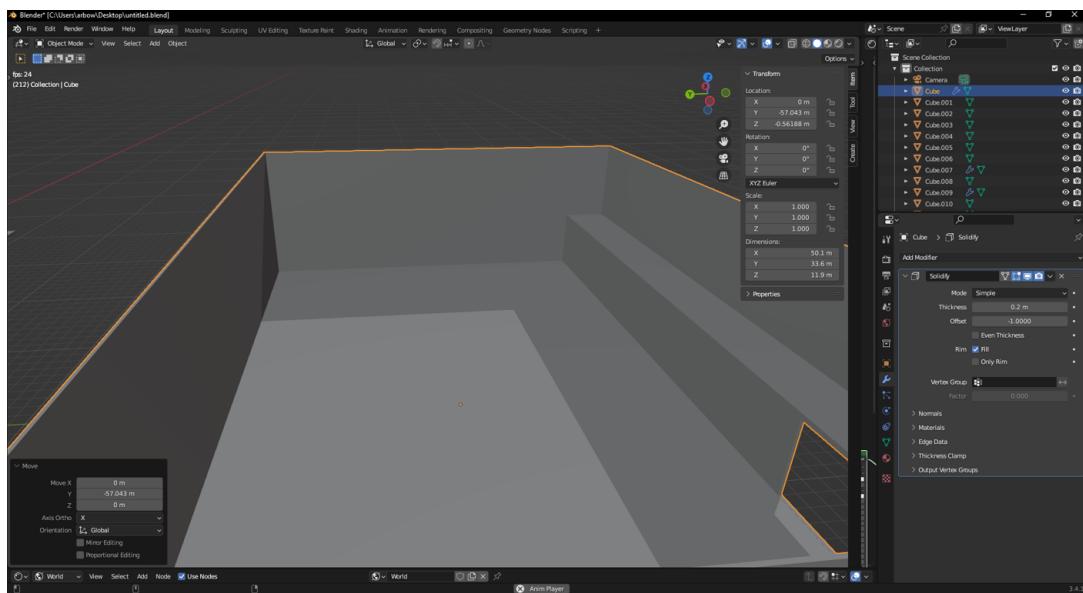
- [https://www.youtube.com/watch?v=xiB5Yu0Shukt=908sab\\_channel=TiGeek](https://www.youtube.com/watch?v=xiB5Yu0Shukt=908sab_channel=TiGeek)
- <https://youtu.be/F8cYVDnuwHs>
- <https://www.youtube.com/watch?v=xmUfhGpA5qA>

#### 3.1.3 Crer un map selon les proportions du btiment C

la contrainte principale tait que la map du mode de jeu devait tre inspire du btiment C. Ainsi, je me suis mis  filmer l'intrieur du btiment et  calculer les dimensions de celui-ci. Le but tait de crer un btiment  plusieurs tages tel que l'architecture rappel celle du btiment C. Le dbut tait plutt simple puisqu'il s'agissait uniquement de poser des cubes aux endroits o les salles se situaient et de

changer leurs dimensions.

Cependant, je vint rapidement à la conclusion que les dimensions du bâtiment C ne sont pas adaptées aux dimensions voulues du mode de jeu. En effet, le bâtiment C mesure environ 50m de long pour 15m de large avec un allé simple entre chaque étage. Or, le but est ici de créer un labyrinthe. J'ai donc décidé de multiplier les dimensions de largeur de ma map par 2(soit 30m de large).



Nouvelles dimensions map



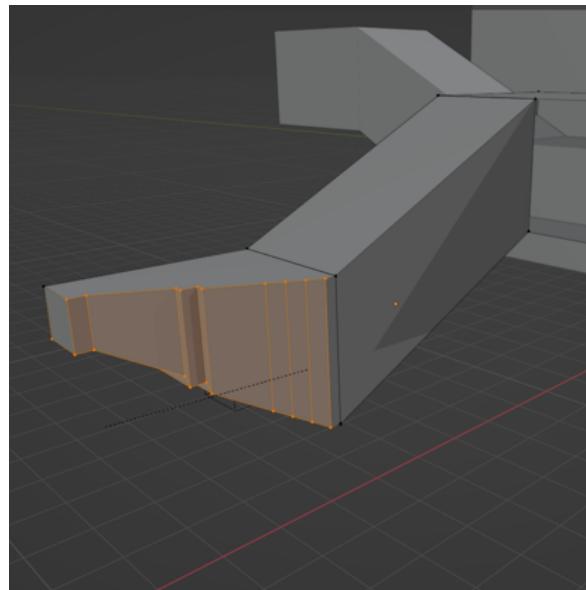
Photo bâtiment C



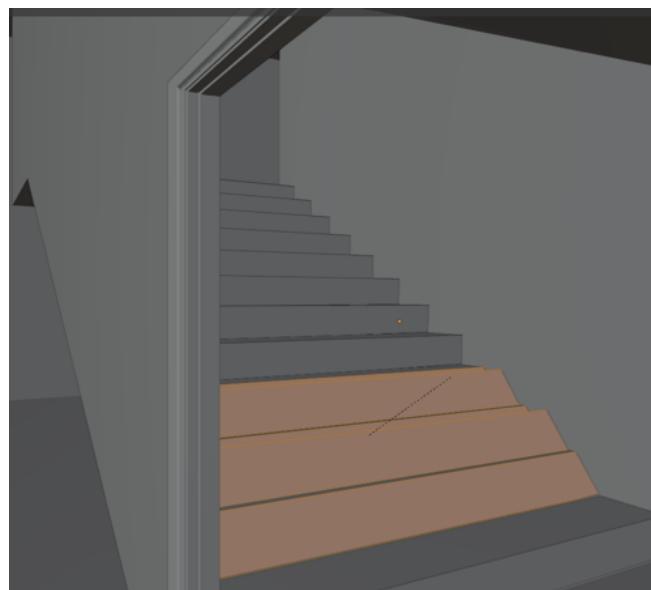
Map du bâtiment C

### Sculpter ses propres formes

Cependant, vint une nouvelle difficulté. Une fois toutes les salles positionnées là où je le voulais, il fallait maintenant sculpter ces salles de manière à obtenir des formes précises. Mais tout cela devait partir d'un simple cube. C'est certainement ce qui a pris le plus de temps dans le développement de la map car j'ai eu beaucoup de mal à créer des formes précises qui correspondent aux dimensions voulues et sans qu'il y ait de débordement sur d'autres objets.



Exemple de sculpture 1

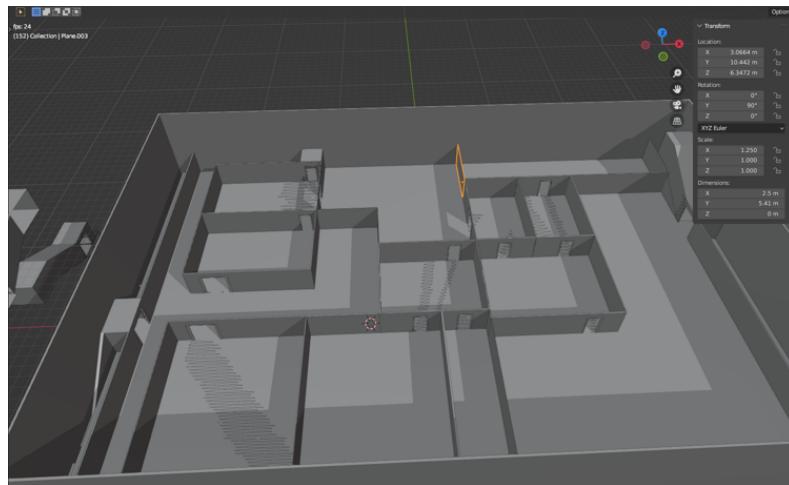


Exemple de sculpture 2

#### 3.1.4 Crédit des niveaux

Une fois tous ces outils pris en main, je pouvais dorénavant me consacrer entièrement à la création d'un parcours technique pour le joueur. Le principe était de

monter d'étages en étages en effectuant des tâches pour atteindre le dernier étage sans se faire attraper par les profs (IA). Ainsi, j'ai positionné les escaliers de chaque côté de la map et j'ai créé un petit labyrinthe pour chaque étage. La map sera toutefois amenée à changer en fonction de nos futures attentes.



Exemple du niveau 1 de la map

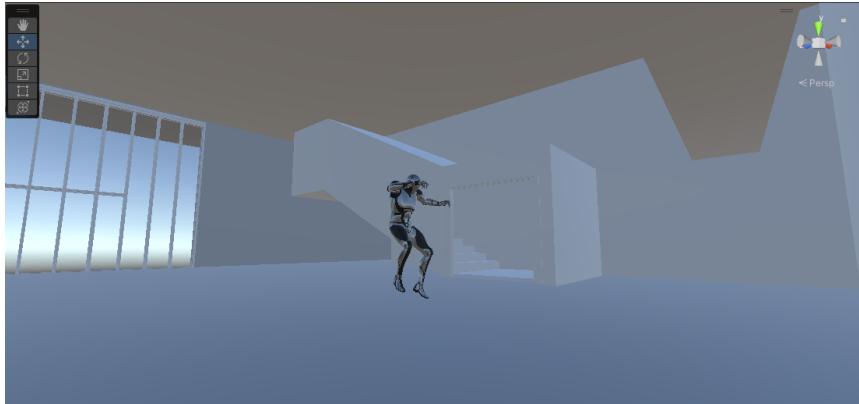
### 3.1.5 Importer mon projet sur Unity

Maintenant la modélisation de la map terminé, il ne me reste plus qu'une étape : L'Importation sur unity. Il faut maintenant importer la map sous format .fbx et ajouter les collisions avec la structure afin que le joueur puisse se déplacer librement sur la carte

Cependant, je rencontre un nouveau problème. Certaines des fonctionnalités que j'ai utilisées lors de la création de la map sous Blender ne sont pas compatibles, ou du moins ne sont pas reconnues par Unity. Par exemple, les trous que j'ai créés dans les murs avec la propriété "boolean" ne sont pas vus comme des trous et toutes les formes que j'ai rendues invisibles sont quand même visibles dans Unity.

Je dois donc retourner sur Unity et trouver un autre moyen de percer mes murs et de créer des murs invisibles. Après plusieurs recherches, j'ai découvert que créer mes propres sculptures avec des trous était la meilleure option, mais cela prenait également beaucoup de temps car il fallait couper chaque sculpture individuellement pour créer les trous souhaités.

Après avoir résolu ce problème et ajouté les collisions à ma map depuis Unity, je peux enfin tester ma map grâce à un asset de personnage sur Unity..



Importation de la map sur unity

### 3.1.6 Retour

C'était la première fois que je me lançais dans la création d'une map de jeu vidéo. Au départ, créer une map selon mes envies me semblait plutôt simple. Cependant, j'ai passé beaucoup de temps à prendre en main les logiciels à ma disposition et j'ai rencontré de nombreux obstacles qui m'ont permis de perfectionner cette map.

Mon objectif pour la première soutenance était simplement de créer une map jouable afin de pouvoir commencer les tests du mode de jeu.

Pour les soutenances suivantes, je remplacerai Léo dans sa tâche de créer les textures de type low poly associées à la map et commencerai à créer les détails de la map pour la rendre plus esthétique. Je serai également chargé de créer notre personnage principal et d'aider mes camarades dans leurs tâches de développement sur Unity.

## 3.2 Deuxième Soutenance

### 3.2.1 Intégration des IA sur la map

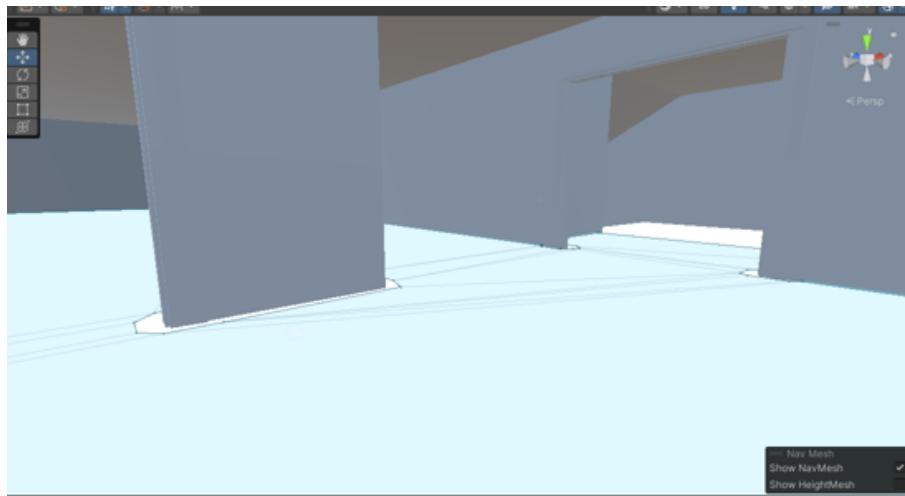
Après la première soutenance, nous avons réparti les tâches de manière à pouvoir rassembler la plupart de nos créations et mettre au point le début de notre jeu. Comme le jeu est essentiellement basé sur le mode de jeu principal (le vol de sujet), notre objectif était donc de construire une base solide pour ce mode de jeu.

Cependant, nous avons également découvert la difficulté du multijoueur, notamment l'intégration et la synchronisation des visuels. Cette difficulté a été particulièrement bien surmontée par Ivan et Talvin, qui se sont occupés d'intégrer les scripts les plus complexes à synchroniser.

Ma première tâche a été d'intégrer l'intelligence artificielle développée par Talvin à la carte. Dans un premier temps, il fallait définir les endroits où l'IA était autorisée à se déplacer et ceux où elle ne l'était pas. Chaque IA apparaît à un certain étage du bâtiment (certains étages où la difficulté est particulièrement élevée possèdent

jusqu'à 2 IA) et les IA ne peuvent pas emprunter les escaliers afin de rester sur leur étage surveillé.

Dans un second temps, il a fallu détecter le joueur en tant que cible et ajouter un script pour que le joueur soit téléporté dans une salle de permanence lorsqu'il est capturé. Il a donc fallu créer une nouvelle salle dans le bâtiment réservée à cet effet.

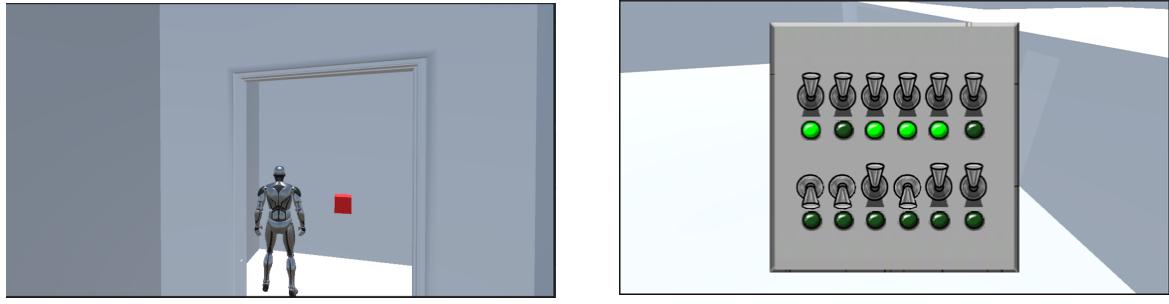


### 3.2.2 Le mode de jeux

Maintenant que l'IA est intégrée à la carte, il faut créer le mode de jeu et le parcours pour voler les sujets du dernier étage. Mes camarades et moi avons donc pensé à l'idée de créer un système de tâches de manière à ce qu'aucun joueur ne puisse accéder à l'étage suivant du bâtiment tant que les tâches liées à l'étage n'ont pas été effectuées.

Pour ce système de tâches, nous avons directement pensé à des mini-jeux similaires à ceux que l'on peut trouver sur Among Us, avec un nombre de tâches imposé à chaque étage afin d'ouvrir la porte et le passage pour l'étage suivant. Ces mini-jeux ont pour but de pouvoir être réalisés en quelques secondes. Cependant, le joueur devra bien calculer sa manœuvre s'il souhaite effectuer la tâche sans se faire attraper par le professeur (l'IA), car lors de l'accès à une tâche, le joueur sera bloqué sur la tâche jusqu'à ce qu'il la termine ou qu'il décide d'arrêter de l'effectuer, auquel cas la tâche se réinitialisera.

Je me suis donc lancé et j'ai créé une première tâche à l'aide d'un canvas. Celle-ci sera à l'avenir ma base pour créer toutes les autres tâches qui seront implémentées dans le mode de jeu.



Bouton accès tâche

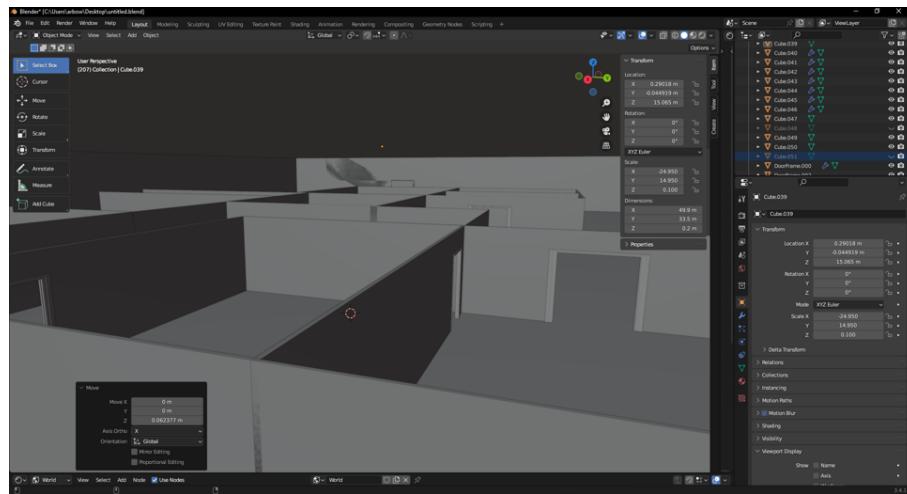
Tâche - Mini jeux

### 3.2.3 Finalisation de la map du mode de jeux

Afin d'avoir un mode de jeu qui puisse être joué de manière logique, il fallait désormais une map correspondant au mode de jeu. Lors de la première soutenance, j'ai créé la map du mode de jeu et j'ai créé provisoirement un petit labyrinthe aux deux premiers étages. Or maintenant que j'ai pu obtenir un petit aperçu de ce à quoi ressemblera notre futur mode de jeu, je suis retourné sur Blender pour finaliser la création de la map afin que chaque étage possède son propre défi.

Une fois les modifications et les ajouts terminés, il me reste deux choses à implémenter sur la map, à savoir les boutons correspondant aux systèmes de tâches et les portes qui seront, provisoirement, des cubes qui alternent entre SetActive true et false pour libérer ou bloquer le passage de l'escalier menant à l'étage suivant.

Pour les boutons correspondant aux tâches, il s'agit de petits cubes rouges trouvables à chaque étage qui, avec un simple clic sur "e" après s'être suffisamment rapproché de celui-ci, lancera le système de la tâche correspondant au bouton. Une fois toutes les tâches effectuées à un étage, cela libérera ses portes.



Maintenant que le fond du mode de jeu est effectué, il ne manque désormais plus que la forme. Mon objectif pour cette soutenance est atteint. Pour la prochaine soutenance, je vais certainement me concentrer essentiellement sur la conception de différentes tâches pour chaque étage, l'ajout des textures ainsi que la création

d'une map globale et d'un lobby afin de finaliser la forme du mode de jeu. Enfin, j'aiderai aussi à l'intégration des items sur la map ainsi qu'au système d'achat d'item.

### 3.3 Soutenance Finale

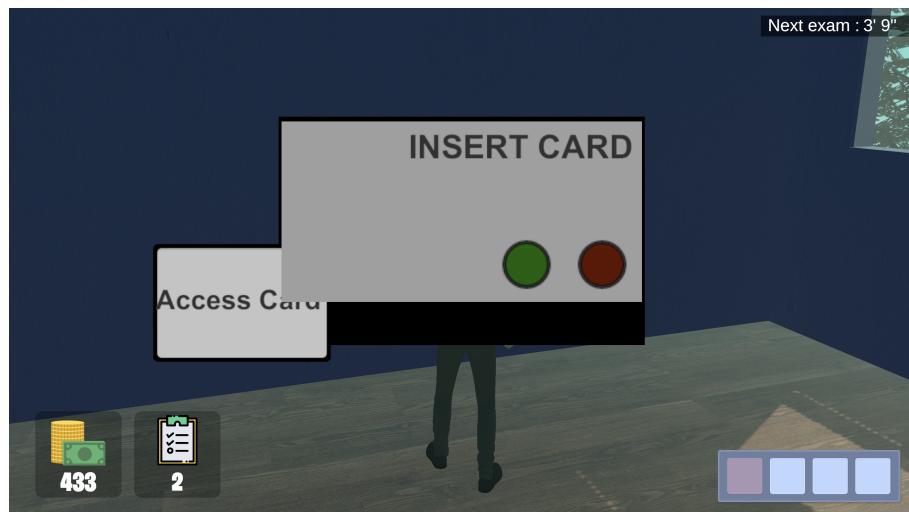
#### 3.3.1 Finalisation du système de tâches et son intégration au mode de jeu

En utilisant le même système de canvas que pour la première tâche, l'objectif était de développer quelques tâches supplémentaires afin de diversifier les activités réalisées par le joueur à chaque étage.

Ces tâches ont été principalement inspirées des tâches d'Among Us, mais ont été modifiées et adaptées pour le mode de jeu. Pour progresser dans le mode de jeu et passer d'un étage à un autre, il est nécessaire d'avoir effectué un certain nombre de tâches dans l'étage actuel, ce qui débloquera l'accès à l'escalier permettant d'atteindre l'étage suivant, à l'image des tâches nécessaires pour gagner dans Among Us. Cependant, ces tâches sont plutôt longues à effectuer et pourraient faire perdre un temps considérable au joueur, car il est immobile lorsqu'il effectue une tâche et ne pourra pas s'enfuir en cas d'arrivée d'un professeur. Il sera donc important de coopérer avec les autres élèves afin d'accomplir les tâches sans se faire attraper.

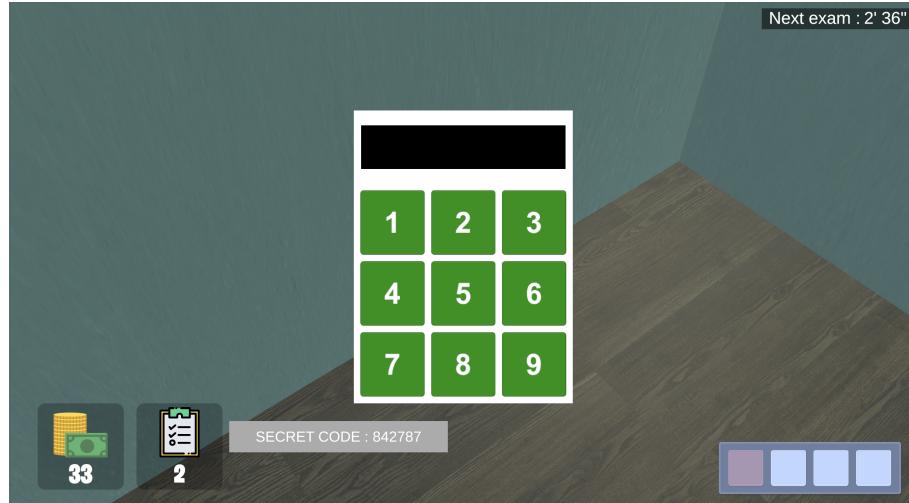
#### 3.3.2 Tâche 2

L'insertion de la carte. La tâche de la carte d'Among Us est particulièrement connue car elle n'est pas simple à effectuer. Il faut glisser la carte à une certaine vitesse ; si cette vitesse est trop rapide ou trop lente, l'accès sera refusé. Cela peut prendre un temps considérable pour les élèves qui essaieraient de dérober les sujets.



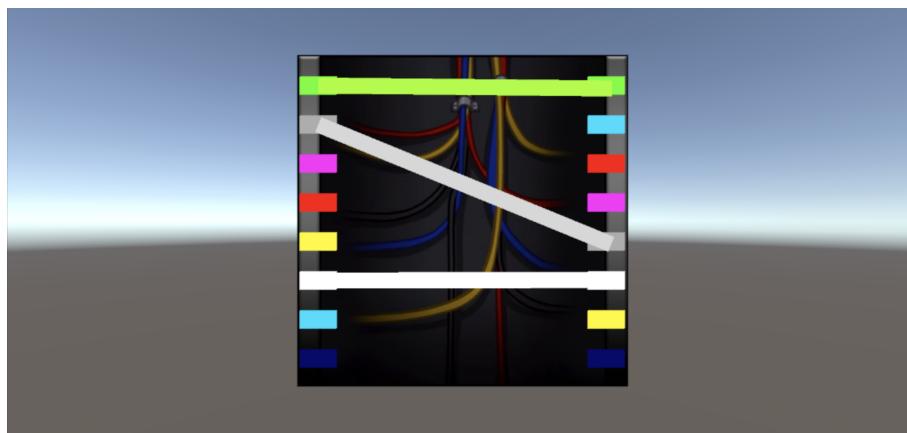
#### 3.3.3 Tâche 3

Code secret. Cette tâche est la plus rapide à effectuer. Il s'agit simplement de taper le code affiché le plus rapidement possible dans l'interface.



### 3.3.4 Tâche 4

Le câblage. Cette tâche, encore une fois inspirée d'Among Us, consiste à relier huit câbles différents entre les entrées et les sorties.



### 3.3.5 Répartition et réinitialisation des tâches

Pour le mode de jeu, nous voulons qu'une tâche aléatoire soit attribuée à chaque bouton (emplacement de tâche). Ainsi, les tâches seront réparties de manière aléatoire dans le bâtiment au début de chaque partie. Ensuite, il faut valider la tâche et la réinitialiser de manière à ce qu'elle puisse être réutilisée par le joueur.

## 4 Kaïs

### 4.1 Deuxième soutenance

#### 4.1.1 Mise en contexte

Arrivé en plein milieu du projet suite à la première soutenance, j'ai alors essayé de prendre connaissance des besoins du groupe. On m'a alors expliqué qu'il était nécessaire de fournir un jeu fonctionnel pour la deuxième soutenance. Mais n'étant

pas encore totalement intégré dans le groupe, je me suis alors proposé pour prendre part à la confection du site internet afin que le reste du groupe puisse totalement se concentrer sur la résolution des différents bugs du jeu estimant ainsi qu'il était encore trop tôt pour moi pour pouvoir les aider dans cette tâche.

Lorsque j'ai commencé à réfléchir sur comment j'allais procédé afin de produire le site le plus qualitatif possible je me suis d'abord penché sur des plateformes tels que Bootstrap ou encore wix qui permettent de crée des sites avec une structure déjà toute faite et des templates accessible à tous. Je me suis alors dit que c'était la solution de facilité et qu'il n'y avait pas grand intérêt à procédé de la sorte. Ainsi, j'ai donc décidé qu'il était plus intéressant pour moi d'essayer de réalisé ce site avec des outils tels que le HTML et le CSS et que ça pouvait alors être une très bonne expérience pour moi.

#### 4.1.2 La conception du site

Etant totalement débutant et en total découverte quant à la production d'un site Internet, j'ai donc était entre autre dans l'obligation d'installer quelques logiciels que je citerai au fil des étapes afin de me faciliter cette tâche. J'ai donc dans un premier temps installé le logiciel Node.js qui est un logiciel qui permet d'exécuter des applications JavaScript et le logiciel Visual Studio Code afin de pouvoir écrire mon code dessus. Pour créer ma première page j'ai eu deux options qui s'offraient à moi. La première étant de créer chaque fichier individuellement et par la suite les relier entre eux et la deuxième options étant d'utiliser un outil qui permet de crée rapidement un projet avec ce qui est nécessaire dessus. J'ai donc facilement opter pour la deuxième solution et j'ai alors décidé d'installer un nouveau logiciel qui se nomme Create React App. J'ai pu alors généré un nouveau projet et installer quelques dépendances (react, react-dom, react-scripts) qui vont ni plus ni moins me permettre d'utiliser des scripts pour démarrer le site, étaindre le site etc, directement sur Visual Studio Code. Je me suis alors retrouver avec plusieurs éléments tels qu'une page HTML et une fonction JavaScript qui va renvoyer du code HTML.

#### 4.1.3 La première page

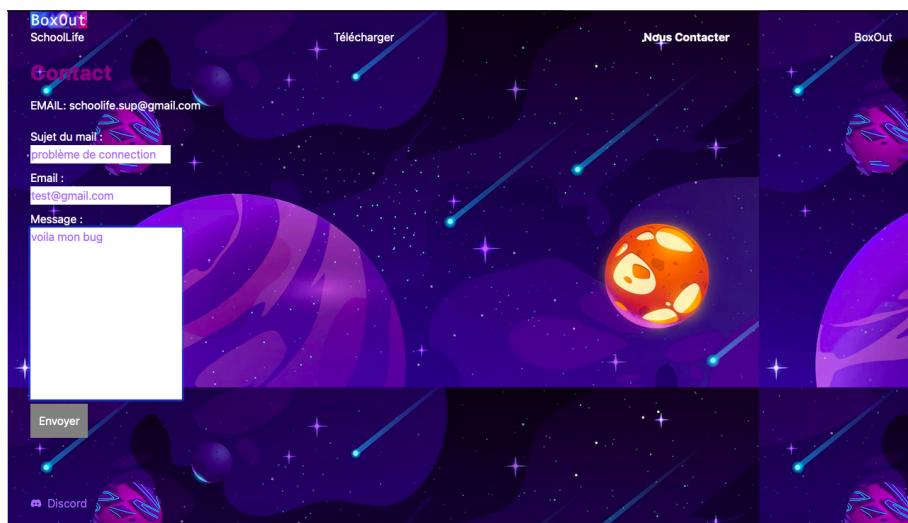
Pour cette première page, j'ai décidé de faire quelque chose de très simple mais qui peut très facilement faire l'affaire. Pour cela j'ai crée un dossier dans lequel j'ai mis toutes les pages de notre site (ici la page d'accueil) et j'ai donc crée un fichier dans lequels j'ai mit une description de notre jeu et une vidéo directement incrusté dans la page qui sera une vidéo trailer/présentation du jeu, pour l'instant c'est juste une video avec un fond noir qui dure dix heures mais pour la soutenance final je prendrais bien évidemment le soin d'upload cette video en privé (ou public à voir) pour pouvoir facilement l'intégré à la version final du site. J'ai aussi pour projet de rejouter une section dans cette page dans laquelle on pourrait regrouper tous les logiciels qui nous auront permis de développer le jeu avec une rapide description de leur utilisation.

#### 4.1.4 La deuxième page

Pour cette seconde page, on peut retrouver l'onglet dans lequelle le joueur pourra télécharger le jeu, j'ai donc simplement mis en évidence un cadre dans lequelle se situe un lien de téléchargement. Pour l'instant ce lien mène vers notre repos git mais lorsqu'on aura fini le développement du jeu je mettrai un lien de téléchargement en .exe pour faciliter la tâche au joueur. Ce qui pourrait être intéressant de rajouter dans cette partie, c'est pourquoi pas une section dans laquelle on mettrai tous les objets, batiments etc, jouables avec lesquelles on mettrai encore une fois une petite description pour faire comme une sorte de guide d'utilisation.

#### 4.1.5 La troisième page

Dans cette troisième page on peut retrouver un formulaire de contact par mail dans lequelle on permet au joueur de nous écrire directement. Le joueur a trois cases à remplir. Dans la première il doit simplement saisir le sujet de son mail, la seconde étant son mail afin qu'on puisse le recontacter en cas de nécessité (cette case doit obligatoirement être remplie sinon le mail ne s'envoie pas) et la troisième étant la case lui permettant d'expliquer son problème, ses recommandation etc. Il m'a semblé nécessaire d'ajouter cette fonctionnalité autant pour le joueur qui pourra nous poser ses questions ou pour nous car cela nous permettra sans doute de recevoir des rapports de bug de la part des joueurs. Cependant pour que cette page aie un sens et qu'elle ne soit pas la juste pour de la décoration, j'ai logiquement du crée une adresse gmail. J'ai notamment utilisé le logiciel EmailJS qui permet de concevoir des mails préfabriqués, j'ai seulement du configurer les placements prédéfinis du mail, du sujet et du message. Ainsi dès qu'un joueur voudra nous envoyer un mail via cette options nous receverons automatiquement un mail. Pour la soutenance finale, j'aimerai pouvoir mettre une sorte de F.A.Q comportant les questions les plus logiques et pertinentes afin de pouvoir ajouter encore une fois une touche de professionalisme dans ce site.





#### 4.1.6 Serveur et Hébergement

Une fois la réalisation de la grande partie du site il faut bien trouver un moyens de l'héberger afin qu'il reste actif tous le temps. Pour se faire j'ai utilisé la plateforme Hostinger. Encore une fois deux options s'offraient à moi, soit j'avais la possibilité d'utiliser un environnement d'hébergement Web soit je faisais le choix d'essayer de tous configurer moi même. J'ai donc fait le choix de le faire moi même et pour cela je suis passé par Hostinger pour mettre en place un VPS et donc d'avoir la possibilité de le configurer à ma manière. Une fois le VPS mis en place il m'a fallut m'y connecter afin de le lier à mon site et pour cela j'ai utilisé le protocole ssh afin d'établir une connexion sécurisé. J'ai cependant rencontré quelques problèmes lors de cette procédure et l'IP du serveur n'est pas totalement bien liée au site et il est donc pour l'instant impossible d'accès pour tous les autres utilisateurs. C'est aussi le cas pour le nom de domaine avec lequel j'ai pour l'instant quelque problème mais bien évidemment j'espere pouvoir régler ces problèmes pour la dernière soutenance.

#### 4.1.7 Design

Pour ce qui est du design j'ai pour l'instant essayé de faire en sorte que le site soit un minimum agréable à regarder et c'est étonnamment la partie avec laquelle j'ai le plus de mal surtout pour les placements des différentes sections dans les pages. La plus grosse partie du design sera cependant présente sur la version finale du site.

### 4.2 Soutenance Finale

#### 4.2.1 L'amélioration du site

Pour cette dernière soutenance j'ai décidé de me concentrer sur la partie design du site web. Donc pour cela j'ai commencé par la page d'accueil dans laquelle j'ai décidé de remplacer la vidéo qui était présente dans le site de la deuxième soutenance par une image du jeu avec le personnage jouable dessus et en fond le paysage du jeu. Par la suite en descendant de la page on peut y retrouver les liens des outils de développement qui nous ont aidé à développer le jeu. Ainsi on peut bien évidemment retrouver Unity, la base du développement du jeu et du multijoueur, Blender qui nous

a permis de modéliser les élément du jeu et Rider qui nous a permis d'écrire notre code.



On peut ensuite retrouver la page qui nous permet de télécharger le jeu. Ainsi l'utilisateur pourra retrouver le lien de téléchargement du jeu, le dossier d'exploitation qui regroupe tous les éléments qui permettent à l'utilisateur de comprendre le fonctionnement du jeu et le rapport de projet qui regroupe tous les éléments expliquant les étapes de développement aboutissant à ce jeu.

Enfin dernièrement on peut retrouver la page de contact qui permet à l'utilisateur de nous contacter en cas de problème.

#### 4.2.2 Player Preferences

Une fois que vous avez quitté la partie vous pouvez être porté à penser que votre argent disparaîtrait dans les abîmes du jeu. Nous avons voulu créer un système de sauvegarde d'argent et de points. Nous avons beaucoup réfléchi à la sauvegarde de nos items, mais nous avons trouvé qu'il serait trop simple « de finir » le jeu en ayant acheté tout les items, la réelle valeur ici sera l'argent, car grâce à celle-ci nous pouvons acheter ce que bon nous semble mais nous ne donnons pas la possibilité de vendre ces objets.

Ainsi, nous sauvegardons l'argent dans les fichier locaux du joueur afin que celui-ci puisse garder sa progression et ne pas perdre son labeur juste à cause d'un crash ou même juste suite à une déconnexion.

## 5 Talvin

### 5.1 Initiation au Projet

Au commencement du projet, j'ai préféré m'initier à Unity par la création de l'Intelligence Artificielle, qui sont deux domaines (Unity et IA) dans lesquels je ne m'étais jamais aventurer. D'ailleurs, l'intelligence Artificielle est une des bases de notre jeu School Life, puisqu'elle représente entièrement le fonctionnement des professeurs.

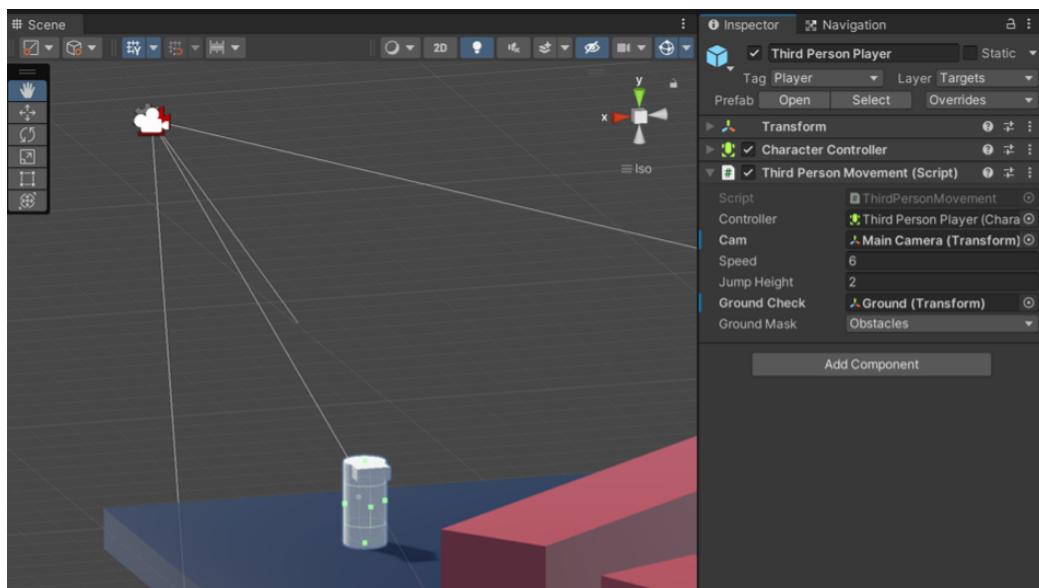
En tant que débutant, je voyais l'Intelligence Artificielle comme quelque chose de très vague, mais également de très compliqué à prendre en main. C'est pourquoi avant de commencer toute création de code ou même de projet Unity, j'ai préféré m'instruire à l'aide de vidéos YouTube, tels que celles-ci :

- <https://www.youtube.com/watch?v=dmQyfWxUNPwt=813s>
- <https://www.youtube.com/watch?v=qOQVxPQ-C5Yt=738s>
- <https://www.youtube.com/watch?v=xmUfhGpA5qA>

## 5.2 Fonctionnement d'un Personnage

Une fois que j'estimais connaître suffisamment mon sujet pour commencer à créer notre propre Intelligence Artificielle, c'est-à-dire notre professeur, j'ai commencé la création de mon projet Unity, sans me douter que mon premier problème serait de savoir comment créer un personnage principal fonctionnel, pour pouvoir tester mes Intelligence Artificielle, et un sur lequel je pourrai les appliquer.

J'ai ainsi commencé par la création de ces personnages sous formes de « cylindre ». Cela m'a pris plus de temps que je ne le pensais, puisque je voulais, avant tout, comprendre ce que je faisais, et non pas recopier intuitivement la création d'un personnage via Internet. Et finalement, je réussis enfin à obtenir un personnage fonctionnel, qui était ma première création Unity, et dont j'étais on ne peut plus fière.



IA Unity

## 5.3 Intelligence Artificielle

### 5.3.1 Cration

 prsent, je pouvais enfin dbuter la cration de mon Intelligence Artificielle. Je commencais donc par reprendre les sources que j'ai pu citer prcemment, puis j'implenta une premire version de notre professeur, grce  laquelle j'ai pu dcouvrir un nombre indenombrable de fonctionnalits Unity, tels que :

- Les Raycasts
- Les Navmesh Agents
- Les Colliders
- Etc ...

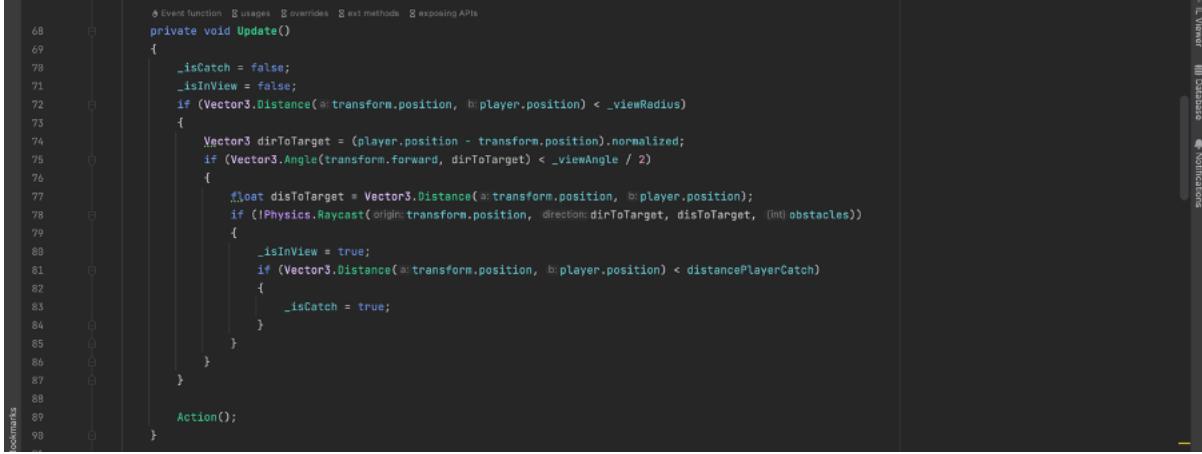
Je parvins alors assez rapidement  obtenir ma premire version de professeur, que j'ai principalement pu crer  l'aide du component «NavMesh Agent».  ce moment-l, mon professeur tait capable de detecter ma prsence lorsque j'tais  une certaine distance de lui, et de se dplacer jusqu' la dernire position  laquelle il m'avait dtect si j'ai pu sortir de son champ de vision, ou bien jusqu' mon personnage s'il russit  m'attraper.

### 5.3.2 Premier problme et solution

En revanche, j'avais galemnt repr les premiers dfauts de mon Intelligence Artificielle. En effet, lorsque je faisais des tests sur celle-ci, ceux-l taient fait sur une surface plane, avec aucun autre objet tels que des murs, ou plus gnralement des obstacles. Ainsi, je n'avais pas pris en compte certains cas, tels que celui ou un mur se trouve entre le professeur, et le joueur, et o dans ce cas-l, le professeur ne devrait pas pouvoir le voir, mais ici, mon professeur le detecte tout de mme.

J'ai donc finalement russi  l'aide de recherches Internet  trouver une solution  ce problme, qui fut entirement rsolu grce  une meilleure utilisation des « Raycast » et aux « LayerMask ». Mais  prsent, cette Intelligence Artificielle n'effectuait pas encore de dplacement de base, c'est--dire de dplacement par elle-mme, mme si je suis pas dans son champ de vision.





```
68     * Event function 0 usages 0 overrides 0 ext methods 0 exposing APIs
69
70     private void Update()
71     {
72         _isCatch = false;
73         _isInView = false;
74         if (Vector3.Distance(transform.position, b:player.position) < _viewRadius)
75         {
76             Vector3 dirToTarget = (player.position - transform.position).normalized;
77             if (Vector3.Angle(transform.forward, dirToTarget) < _viewAngle / 2)
78             {
79                 float disToTarget = Vector3.Distance(transform.position, b:player.position);
80                 if (!Physics.Raycast(origin:transform.position, direction:dirToTarget, distance:disToTarget, (int)obstacles))
81                 {
82                     _isInView = true;
83                     if (Vector3.Distance(transform.position, b:player.position) < distancePlayerCatch)
84                     {
85                         _isCatch = true;
86                     }
87                 }
88             }
89         }
90         Action();
91     }
```

Code de la vue de l'IA

### 5.3.3 Finalisation

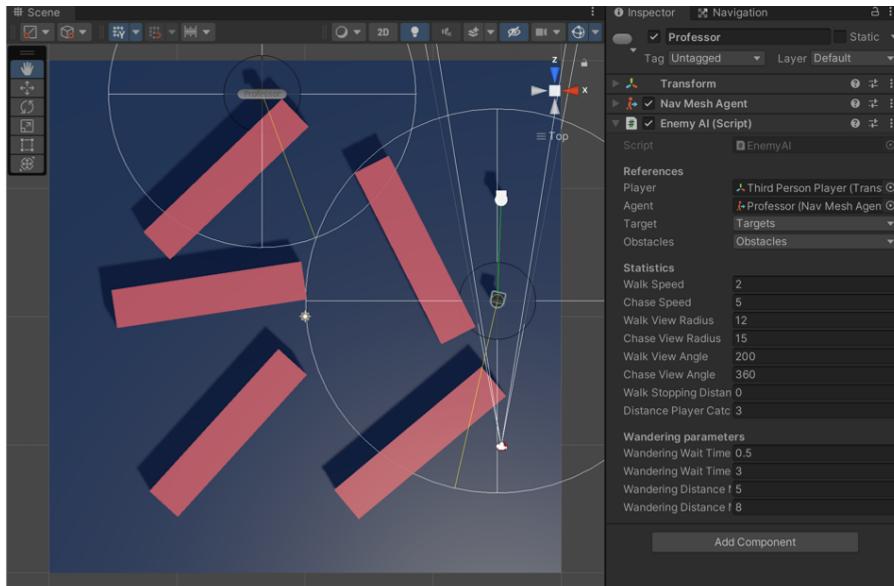
Par la suite, j'ai donc ajouté deux « états » à mon professeur :

- État de PNJ / lorsque le professeur se balade simplement dans les couloirs ;
- État de poursuite / lorsque le professeur est en train de poursuivre un élève.

Grâce à ceux-là, le professeur peut poursuivre un élève s'il en repère un, ou sinon faire sa vie de professeur.

Seulement après, j'ai commencé à ajouter des caractéristiques dynamiques à mon Intelligence Artificielle, tels qu'un champ de vision, une distance de vue, une vitesse de déplacement etc... Ces éléments comme j'ai pu le dire sont dynamiques en fonction de l' « état » dans lequel est notre Intelligence Artificielle. Par exemple, lorsque le professeur marche simplement dans un couloir, sa vitesse de déplacement est « normale », comparé à lorsqu'il poursuit un élève, où ici, elle est « élevée ».

Enfin, mon Intelligence Artificielle est finalisée pour le moment, et fonctionne presque exactement comme nous l'imaginions.



### 5.3.4 Import des IAs sur le Multijoueur

Enfin, mon dernier devoir lié au fonctionnement des Professeurs était de les faire fonctionner sur le Multijoueur, et de leur donner un visuel plaisant (autre qu'un cylindre).

Pour les intégrer au multijoueur, j'ai d'abord essayé de les faire fonctionner sur la base d'un Client Authority (comme on l'a fait pour absolument tous les autres éléments de notre jeu), mais qui pour les professeurs ne pouvait pas fonctionner. En effet, chaque client avait une position différente du professeur, car chaque client instanciait une position différente aux professeurs, et donc ils n'étaient pas coordonnées sur le Multijoueur.

Ainsi, j'ai fais fonctionner mes professeurs sur la base d'un Server Authority, qui consiste à ce que le Server calcul les différents endroits où les professeurs doivent se diriger, et envoie ces informations au client afin qu'il bougent également les Professeurs aux endroits correspondants.

## 5.4 Inventaire

Une fois que la grande partie de l'Intelligence Artificielle était faite, j'ai pu commencer à m'intéresser à l'interface direct de notre joueur. C'est-à-dire à son Inventaire, et aux Items qu'il pourra utiliser. Je vous le rappelle, le but du jeu est de passer l'année, en réussissant toutes les épreuves, et les Items sont là pour nous aider à réaliser cet objectif.

### 5.4.1 Référencement

Tout comme pour l'Intelligence Artificielle du jeu, je ne savais absolument pas comment faire pour coder un Inventaire, mais cela ne me semblait pas très compliqué. J'ai donc commencé regarder des vidéos YouTube pour encore une fois m'instruire

à ce sujet. Finalement plusieurs vidéos ont attiré mon attention, et j'ai donc à ce moment décidé de m'inspirer de ces plusieurs vidéos pour construire mon inventaire :

- <https://youtube.com/playlist?list=PLUWxWDlz8PYIvzRTHyvx54tgWTCKgZzkZ>
- <https://www.youtube.com/watch?v=AoDf1fSFFg>
- <https://www.youtube.com/watch?v=ZYG1Db7GiZM>

#### 5.4.2 Création

J'ai alors commencé la création de mon inventaire. Et je me suis rendu compte que c'est bien plus difficile que ce à quoi je m'attendais, et bien plus compliqué que la création de l'Intelligence Artificielle.

Durant son développement, j'ai créé une multitude de Scripts, j'ai rendu possible à mon personnage de détecter si élément est devant lui avant qu'il puisse le ramasser, j'ai crée mon inventaire sous forme d'une liste, je fais correspondre ses éléments aux slots de l'inventaire, etc..

#### 5.4.3 Encore des problèmes...

D'ailleurs, pour parler de mes peines durant sa création, la fonctionnalité de détection de l'objet par le personnage a été une des plus compliqués à créer. Je ne trouvais, à ce moment-là, aucune référence sur YouTube ou autre de laquelle je pourrais m'inspirer, et qui correspondais à ce que je voulais exactement comme fonctionnement. Je cherchais à pouvoir ramasser les objets de la même manière que dans les jeux Call Of Duty : Warzone, Fortnite, PUBG, etc... Actuellement, je pense avoir un script fonctionnel qui est presque à la hauteur de mes pensées.

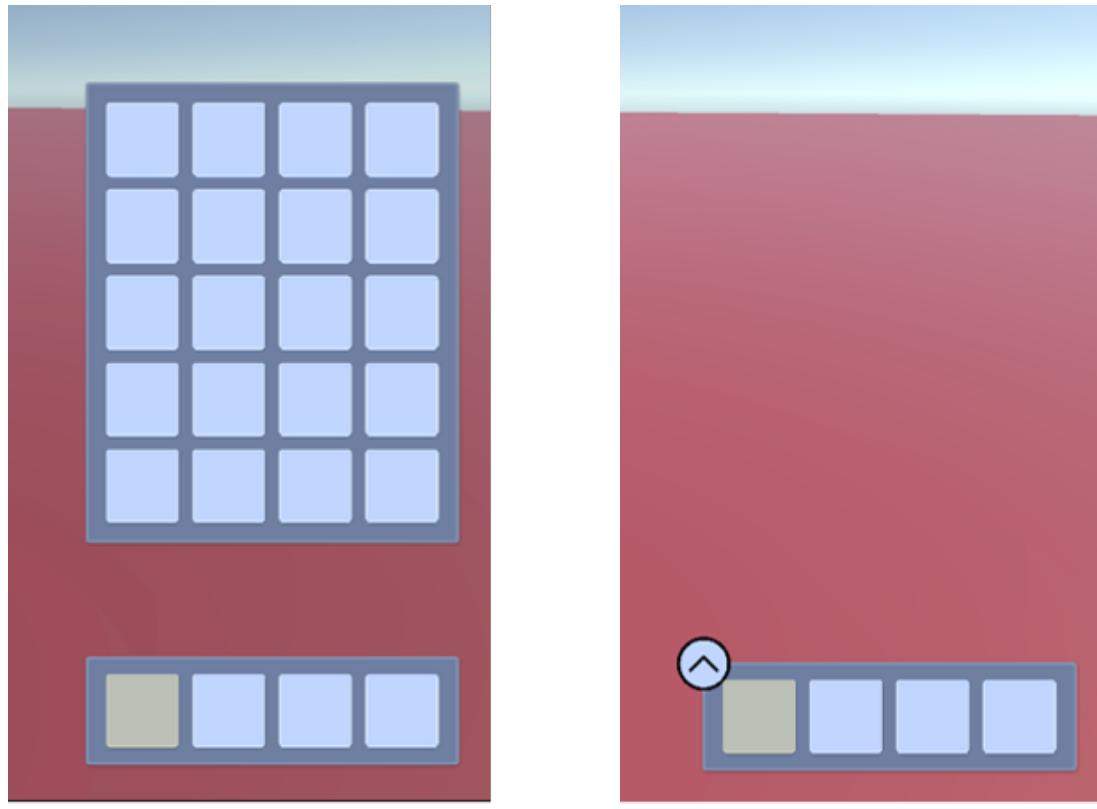
En effet, je ne pouvais pas simplement tirer un RayCast venant du milieu de mon écran vers devant, car cela impliquerait que nous devions visés à l'aide d'un curseur exactement où l'Item se situe, afin que l'interaction du Raycast à ce moment-là nous transmettre le RaycastHit de l'item. Il n'était pas non plus possible de simplement utiliser un SphereCastAll ou BoxCastAll, car si nous avons plusieurs Items de détectés, comment savoir lequel nous souhaitions prendre ?

Finalement, j'ai réussi à créer cette fonction, en effectuant des test par milliers, afin d'obtenir quelque chose de fonctionnel et à la hauteur de mes attentes, avec une utilisation de Physics.SphereCastAll, et de plusieurs Physics.Raycast, Vector3.Angle, Vector3.Distance.

#### 5.4.4 Partie 2D | Visuel

J'ai également découvert l'utilisation de l'interface 2D de Unity, puisque l'inventaire est une fonctionnalité 2D présente sur notre jeu, qui actuellement utilise par

exemple des Boutons, des Texts, des Images, etc. Mais, la création de l'inventaire a été pour moi très compliqué, surtout en vue du fait que je n'arrivais jamais à créer un inventaire regroupant toutes les fonctionnalités dont j'avais besoin. À vrai, dire la fonctionnalité que je voulais absolument, mais qui m'a le plus compliqué la tâche est le système « Drag and Drop » de l'inventaire.



Inventaire Ouvert

Inventaire Fermé

#### 5.4.5 Ressenti Personnel

J'ai remarqué qu'un inventaire dans un jeu vidéo demandait beaucoup plus d'outils et de travail que l'on ne l'imagine. Lorsqu'on joue à un jeu vidéo, on ne remarque pas le travail fait derrière l'affichage du contenu de la case de l'inventaire, la bulle (tooltip) permettant de renseigné ses informations, le "Drag and Drop" système, le slot que l'on utilise actuellement etc..

Actuellement, j'ai tout de même réussi à avoir un Inventaire fonctionnel, après la création de 2 inventaires qui me laissaient insatisfait. Certes ce travail était plus compliqué que ce que je n'imaginais, mais il m'a permis d'en apprendre plus sur Unity, les algorithmes et C en lui-même.

### 5.5 Items

J'aimerais à présent vous faire une présentation assez simple de chaque items que jai pu créer (tous sauf le Casque étant fait pas Ivan), sans pour autant rentrer

dans les détails de leurs implémentations.

Voici les Items que nous avons ainsi ajoutés à notre jeu :

- Clé : Donne accès à des salles fermées.
- Téléphone : Émet un son et attire les professeurs.
- Lunette : Permet de voir pendant 10 secondes les professeurs à travers les murs.
- Sujet : Permet d'obtenir 20 au QCM actuel.
- Batte : Permet d'assommer un élève pendant 5 secondes (ne peut pas être utilisé contre un professeur).
- Casque : Permet d'écouter de la musique.

## 5.6 Intégration des tâches au Multijoueur

Ensuite, en ce qui concerne des tâches, ayant été faites par Titouan, mais Titouan ne s'étant pas introduits au système du multijoueur, j'ai intégré chacune des tâches à notre jeu et au Multijoueur, afin que chaque joueur puisse réaliser ses tâches personnellement.

J'y ai également ajouter un système empêchant un joueur de réaliser plusieurs fois une même tâche, afin que : si la tâche correspond à un cours, il ne puisse pas gagner de point trop facilement, sans rien faire, et si la tâche correspond à une nous permettant de voler le sujet, il ne puisse pas récupérer toutes les clefs permettant l'accès aux étages, facilement.

En revanche, un problème a eu lieu lors de l'import des tâches. Kaïs et Titouan ayant chacun fait des tâches, indépendamment du projet commun, certaines d'entre-elles étaient réalisés en fonction de la caméra du joueur. Or, il m'était ainsi très difficile de l'intégrer au Jeu car la caméra n'est pas immobile, et d'autres éléments dans la scène seraient susceptibles de passer entre la caméra et la tâche effectuée.

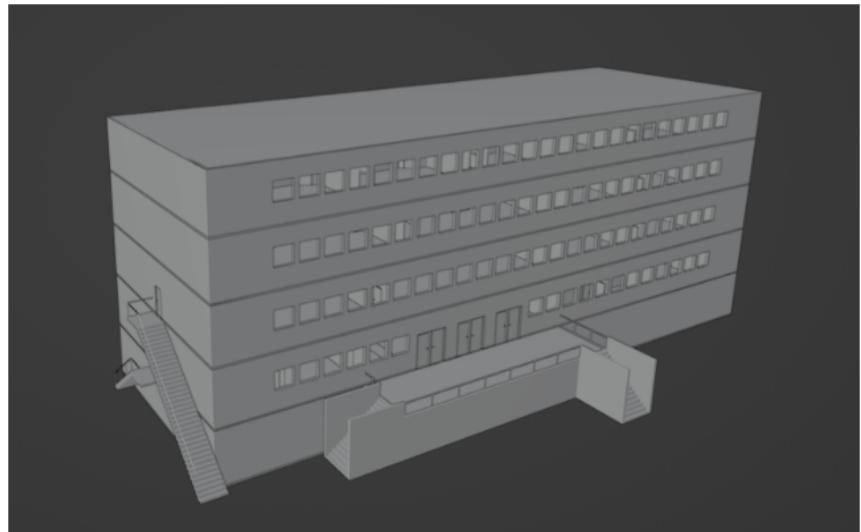
## 5.7 Batiment A

### 5.7.1 Modélisation

Je vais à présent vous faire part du dernier travail que j'ai fourni vis-à-vis de ce projet, qui est la construction complète du bâtiment A. Comment vous dire que c'était la tâche la moins plaisante que j'ai pu réaliser jusqu'à présent ? Mais étant quelqu'un de très perfectionniste, j'ai tout de même eu l'intention de réaliser un bâtiment correspondant aux réelles proportions et de la meilleure des manières.

Je me suis donc initié à Blender, je m'y suis donc instruit comme à chaque fois en regardant des vidéos YouTube, et une fois que je connaissais le mécanisme global de cet application, j'ai pu commencé sa création. À vrai dire, la construction des murs et de l'architecture global du bâtiment n'était pas si compliqué. Ce sur quoi je rencontrais le plus de difficultés était en réalité les murs courbés, présents aux

1er, 2eme, et 3eme étages, ainsi que tous les escaliers présents. J'estime tout de même avoir passé une cinquantaine d'heures dessus, et cela sans compter l'ajout des textures au bâtiment.



Model du Batiment A

### 5.7.2 Textures

Après avoir fini la modélisation complète du bâtiment, je me suis donc attaqué aux textures de celui-ci. Pour celà j'ai recherché partout sur le Web, ne connaissant pas encore de sites fait pour cela, sans trouvé de textures très convaincantes, et c'est alors qu'Ivan me conseilla le site Qixel, grâce auquel j'ai pu y retrouvé toutes les textures dont j'avais besoin. J'ai ainsi pu les appliquer sur mon model, et le bâtiment A était enfin finie.

Mais, il restait tout de même encore un problème dont je m'attendais le moins, qui était simplement que lors de l'import de mon model vers Unity, il était impossible de garder les textures qui y étaient déjà appliqués. Cela m'avait extrêmement surpris que cet fonctionnalité n'existe pas, et j'ai donc passé 2 jours entiers à essayé de résoudre ce problème depuis Blender, pour au final importer les textures puis Unity même.



Textures du Batiment A