# PROJET S2

---

# Nacho's Rumble

Rapport première soutenance

---

Los Successos


Paul Berthelot (chef de projet)

Théo Charbonnier

Kilian Hardy

Sam Esber


Mars 2022

# Table des matières

# 1 Introduction

Los Successos a le plaisir de vous présenter *Nachos Rumble!* Un jeu de tir à la première personne à l'ambiance mexicaine et au gameplay simple et amusant. Depuis le début du projet notre groupe est composé de Paul, Sam, Kilian et Théo. Nous avons beaucoup appris durant cette première partie de développement, notamment sur la manière de travailler correctement en équipe, ce qui est absolument nécessaire pour la suite du projet.

Nous avons pris en main les logiciels nécessaires au développement du jeu que nous avons cité dans le cahier des charges ainsi que des outils permettant au groupe d'être organisé et efficace.
Nous sommes contents du déroulement de ce projet car nous le voyons se construire doucement mais sûrement, et on en serait presque impatient d'en voir le résultat.

Il reste encore beaucoup de chemin à parcourir avant d'arriver à un jeu fini mais nous sommes déterminés à redoubler d'effort.

## 1.1 Modification du cahier des charges

Le planning prévisionnel des tâches à effectuer à été un peu modifié :

- Réseau: 20% → 60% La gestion du réseau s'est avérée plus simple que prévue sur certains points tels que l'instantiation des salles de jeu et les interaction entre joueurs.
- Interface: 40% → 20%
  Nous avons décidé de réduire l'avancement prévu de la partie Interface, car nous avons considéré plus important d'implémenter les mécaniques du jeu en lui en même en privilégiant la fonctionnalité à la beauté.

## 1.2 Résumé du travail effectué

Le but du travail à faire pour cette première soutenance était de poser clairement et proprement les bases de notre jeu et ses axes de développement. C'est pourquoi avant de travailler sur l'aspect visuel, nous nous sommes focalisés sur la mise en place des mécaniques les plus importantes d'un FPS: les déplacements, le système de tir et de dégâts, le système de parties multijoueurs, etc.

Dans l'état actuel du jeu, les mécaniques de déplacements et le système de base de tir est fonctionnel.
Nous avons créé des menus très basiques pour permettre de commencer le développement plus facilement: des menus simples pour créer et rejoindre des parties en lignes, un menu de réglages pour modifier l'apparence de son jeu (et autre).

Pour faire clair: nous avons posé les bases nécessaires au bon déroulement de ce projet, tout en menant des recherches pour savoir où se diriger dans la suite du développement du jeu. Grâce au travail réalisé jusqu'à maintenant, nous sommes totalement prêts à entamer une phase de travail plus précis et concret, sur l'aspect du jeu, sur ses fonctionnalités plus particulières, etc.

# 2 Avancement des tâches

Dans cette partie, nous abordons l'avancement des tâches de chacun dans la création du jeu de manière individuelle, même si en réalité, nous avons la plupart du temps travaillé à plusieurs pour s'aider et rendre l'avancement plus rapide et mieux réalisé.

## 2.1 Paul

### 2.1.1 Instantiation des salles de jeu

Pour gérer les interactions multijoueurs de notre jeu, nous avons décidé de se servir du service de Photon, qui permet gratuitement d'utiliser des serveurs avec jusqu'à 20 joueurs en simultanés.
Photon est très bien intégré à Unity et comme première tâche du projet, j'ai réalisé des menus très simples permettant aux joueurs se connectant au jeu de créer des salles de jeu ou de les rejoindre. Seule la personne ayant créé la salle de jeu est ensuite en capacité de lancer la partie. J'ai implémenté la gestion de migration d'hôte (si le joueur ayant la possibilité de lancer la partie quitte la salle ou le jeu) qui redonne les droits sur la salle à un autre joueur.

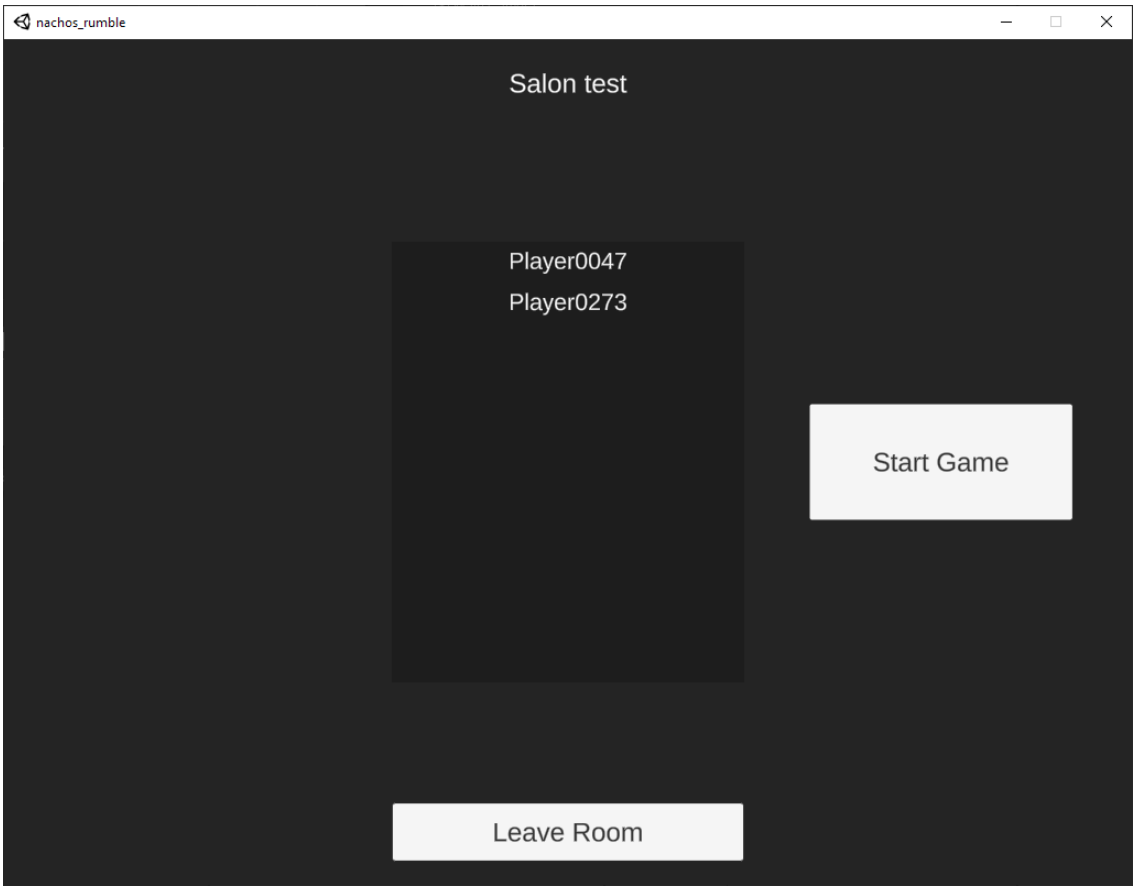


Figure 1: Interface: liste des joueurs dans une Room

### 2.1.2 Caméra

En tant que *FPS*, la caméra des joueurs de Nachos Rumble est évidemment placée au niveau des yeux du personnage joué. J'ai implémenté la gestion de la caméra, chose basique mais qui doit être bien faite étant donné que l'ensemble du jeu repose sur cette simple fonctionnalité (notamment les mécaniques de tir). La sensibilité de la souris pour faire bouger la vision du joueur est modifiable dans le menu des réglages du jeu. Il était aussi important de limiter la vue du joueur verticalement, c'est à dire de ne pas pouvoir regarder plus bas encore quand on regarde droit vers le sol, ou plus haut quand on regarde droit vers le ciel. Il a aussi fallu gérer le fait de ne plus affecter la caméra lorsque le joueur est dans un menu (par exemple celui de réglages, on ne veut pas que la vue du joueur bouge lorsque celui-ci choisit ses options).

### 2.1.3 Mécanique de tir

Avec l'aide de Kilian, nous avons implémenté les bases des mécaniques du tir du jeu. C'est à dire que lorsque le joueur a une arme, le jeu instancie le modèle 3D de l'arme, avec laquelle il peut tirer, et donc infliger des dégâts aux joueurs de l'équipe ennemie. Le tir en lui même est généré par un script. Il s'agit d'un "Raycast", c'est à dire une ligne droite instantanée, partant du centre de la caméra du joueur. C'est un moyen simple et très utilisé d'implémenter un système de tir dans un jeu, cela évite par exemple plusieurs problèmes lorsqu'on essaie de faire partir le tir du canon de l'arme.
Ce raycast permet de récupérer les informations sur les cibles touchées, comme par exemple leur points de vie, et quand ceux-ci sont inférieurs ou égaux à zéro, le joueur meurt. Les points de santé de chaque joueur sont stockés dans un script propre au joueur. Nous n'avons pas encore perfectionné le système de tir, chose qui viendra par la suite évidemment, il manque par exemple le fait de rajouter de la dispersion au tir (du hasard dans la répartition des balles tirées), implémenter du temps de parcours des balles pour certaines armes ou encore créer l'impact des balles sur le décor du jeu, voire même d'autres moyens de tirer des balles, comme en créant des projectiles.
En effet, nos armes sont toutes gérées par leurs propres scripts contenant des informations sur leurs fonctionnements (munitions, dégâts, cadence de tir, etc), ce qui nous permet facilement d'implémenter de nouvelles armes. Il s'agit d'ailleurs de quelquechose que nous essayons de généraliser à l'ensemble du projet: faire des scripts qui permettent de gérer d'autres scripts dans le but de rendre la réutilisation du code dans le futur bien plus simple, rapide et sans prise de tête.

### 2.1.4 Multijoueur

Les interactions entre les joueurs sont assez faciles à implémenter avec Photon. Il a été assez rapide de mettre en place le système de création et lancement de room. Nous allons très sûrement rester sur le système actuel de room pour notre multijoueur, étant donné qu'à l'échelle de ce projet, proposer un matchmaking (recherche de joueurs du même niveau pour former une partie) n'a pas vraiment de sens.

Photon est aussi très utile pour instancier des scripts ou objets 3D, comme les joueurs, les scripts gérant les joueurs et la partie.

## 2.2 Sam

### 2.2.1 Audio

Dans un fps comme le notre, de très nombreuses informations sont transmises par le son, (coup de feu, bruits de pas...)
Durant mes recherches j'ai trouvé de nombreux bruitages déjà disponibles gratuitement sur internet. J'ai fait une pré-selection, de différents sons qui correspondent à ce que l'on attend d'un fps low poly "fun" et il s'agit maintenant de les attribuer, en fonction de la map, des armes, des dégats reçus, etc. J'ai également choisi de selectionner pour l'instant de légers bruitages pour le menu (pour les clics sur chaque bouton et au lancement de la partie).
Il reste encore beaucoup de bruitages précis qui manquent a l'appel (changement d'arme par exemple). Mais dans une généralité, tout se déroule bien.
On s'attendait à ce que cela soit plus dur de trouver autant de bruitages sans qu'ils ne paraissent trop réaliste.

La musique bien qu'accessoire en comparaison à de nombreuses fonctionnalités, garde tout de même une certaine importance, notamment dans le menu, en tant que musique d'ambiance. Mais ce n'est pas tout il y a également les musiques à la fin d'une manche, selon la défaite ou la victoire du joueur.
Il y a eu beaucoup de discussions sur la musique à choisir, une très typique mexicaine quitte à rentrer dans le cliché, ou rester assez sobre avec une musique calme et pas vraiment indentifiable comme hispanique. J'ai finalement choisi de faire un mix des deux, une musique calme mais qui reste identifiable, pour l'ambiance. Et pour les autres musiques, il faut qu'elles soient facilement reconnaisables, par exemple si on gagne, il faut facilement le comprendre. Selon le temps disponible, j'envisagerai de créer ces musiques en partie par moi-même ou bien de trouver des ressources disponibles sur internet.

### 2.2.2 Modèles 3D

Les modèles 3D nécessaires à notre jeu sont les suivants: les personnages, les armes, la map (le décor). Nous avons estimé que le temps consacré à de la modélisation 3D n'était pas le temps le mieux utilisé, et qu'il vallait mieux à la place, implémenter les mécaniques fondamentales de notre jeu. C'est pourquoi nous avons choisi des assets 3D d'armes déjà réalisées sur internet. Le résultat obtenu par nos propres efforts aurait été moins agréble visuellement, et nous y aurions pris du temps plus utile ailleurs. Il s'agit de modèles low poly s'intégrant à la perfection à l'univers et à la touche artistique de notre jeu, nous étions donc ravis de les trouver.

Figure 2: Un des modèles 3D d'armes que notre jeu: *l'AK47*

Quant au personnage, je me chargerai d'en modeliser un (ou plus) correspondant parfaitement à l'esprit mexicain du jeu, pour raffirmer l'identité de celui-ci.

Enfin pour la map, nous sommes très friands de jeu FPS stratégiques (même si notre jeu n'a pas forcément pour cible d'en être un), dans lequel les maps ont une importance majeure. Ce sont les maps qui permettent ou non aux joueurs de s'épanouir en jouant, de créer des stratégies, de réapparaître normalement (ni trop proche ni trop loin du combat), etc.

Nous allons créer nous même la map avec soin, au risque de n'en faire qu'une seule que l'on trouve agréable à jouer et bien faite, plutôt que deux moyennes. Voici un plan (approximatif) de la façon dont on l'imagine, avec en rouge les obstacles derrière lesquels se cacher et en jaune les zones surélevées.

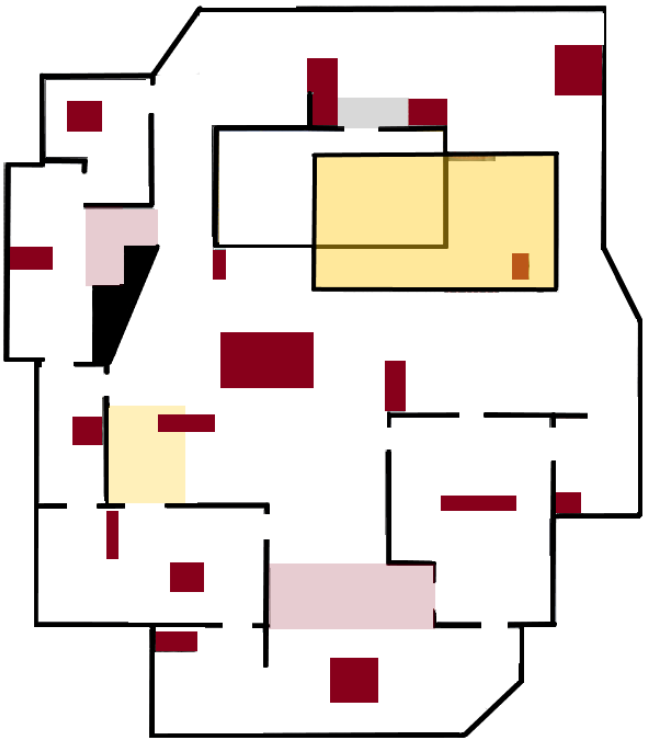

Figure 3: Plan prévu de la map du jeu

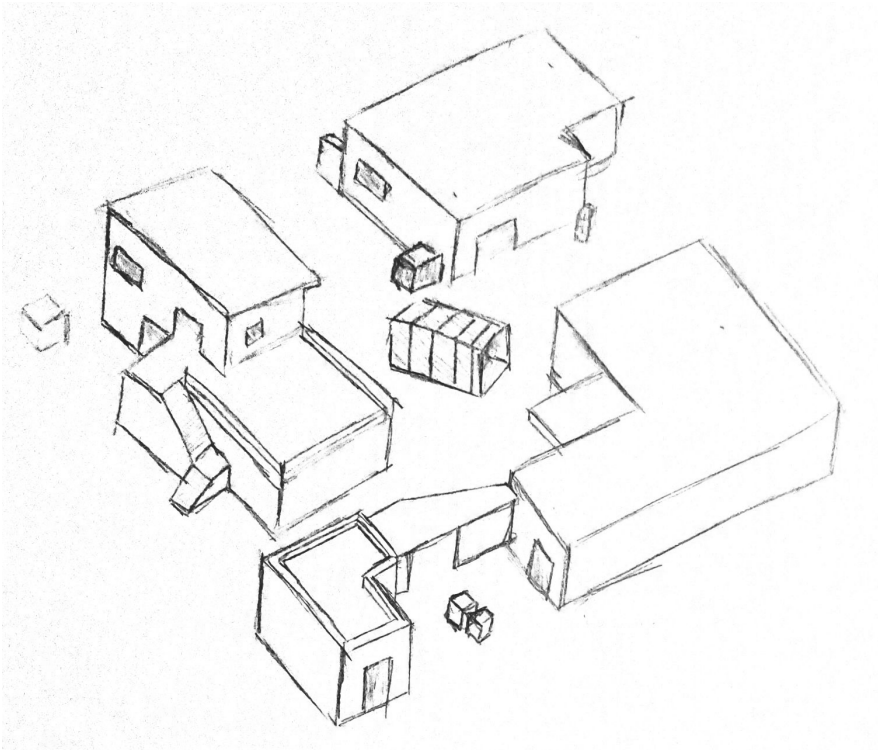

Figure 4: Sketch montrant une vue de ce à quoi pourrait ressembler la map

## 2.3 Kilian

### 2.3.1 Gestion de la partie et des joueurs

Les informations qui sont importantes au déroulement de chaque parties doivent être stockées quelquepart.
C'est pourquoi nous avons utilisé des scripts pour gérer les rooms ainsi que pour chaque joueur d'une partie, qui permettent ici de stocker les données qui leurs sont liées.

Le script *RoomManager* va être généré une seule fois par partie: à son lancement. C'est ce script qui va permettre d'instancier tout ce qui se passe durant la partie, comme par exemple de générer un leaderboard (tableau de classement des joueurs selon leurs performances), gérer l'arrêt de la partie, etc. C'est aussi ce script qui instancie pour chaque joueur présents dans la partie leur scripts *PlayerManager*.

Ce script est celui qui va stocker les données de chaque joueurs: ses points de vie, son état (en train de courir, marcher, sauter, tomber, etc), mais aussi gérer le respawn (les réapparitions sur la map après être mort), et très important: recevoir et envoyer des informations aux autres joueurs (dégâts, etc).
Il va aussi instancier tout les prefabs liés aux joueurs: leurs armes, les colliders utilisés dans la physique et les déplacements, etc.

### 2.3.2 Physique et mouvement

La gestion des mouvements des joueurs est une des mécaniques principales du jeu avec le système de tir.
Pour effectuer les déplacements, nous avons utilisé des colliders. Les colliders permettent d'appliquer des forces afin de permettre les déplacements des objets ou des joueurs, qui pour notre cas est un personnage. Nous avons ensuite appliqué un BoxCollider qui permet de délimiter une zone autour de notre objet délimitant la zone de collision de l'objet dans l'espace, comme certains éléments du décor ou encore les armes, ce qui permet de pouvoir choisir avec quel éléments on souhaite intéragir.

A partir de ça j'ai pu commencer à m'occupé de la gestion des déplacements de notre joueur, ce qui est un peu la base du jeu.
La première étape était de pouvoir faire des mouvements de gauche à droite et de l'avant vers l'arrière, c'est à dire sur les axes X et Y, en utilisant les touches du clavier initialement connues pour ce procéder (Z;Q;S;D). J'ai également créé une variable vitesse, qui définit la vitesse de déplacement du joueur qu'on pourra modifier à notre guise par la suite comme par exemple si l'on souhaite ajouter des bonus dans notre jeu ou changer la vitesse en fonction de l'arme qu'on a en main. Le problème que j'ai rencontré sur cette partie est que mon personnage exécutait très bien ses mouvements mais ne pouvait pas monter des escaliers ou des rampes, ce qui est un handicap si nous avons une map avec des étages. Une fois ce problème réglé, j'ai du implémenté la gravité, parce que sans celle-ci notre joueur resterait en lévitation dans les airs. Afin que ce soit le plus naturel possible, et que ça ne soit pas notre

personnage qui tombe d'un coup sur le sol, il a fallu que je fasse en sorte que le personnage chute de plus en plus vite en definissant une vitesse de chute et que j'augmente au fur et à mesure de la retombée, et que je réinitialise à zéro dès que le joueur est de nouveau en contact avec le sol.

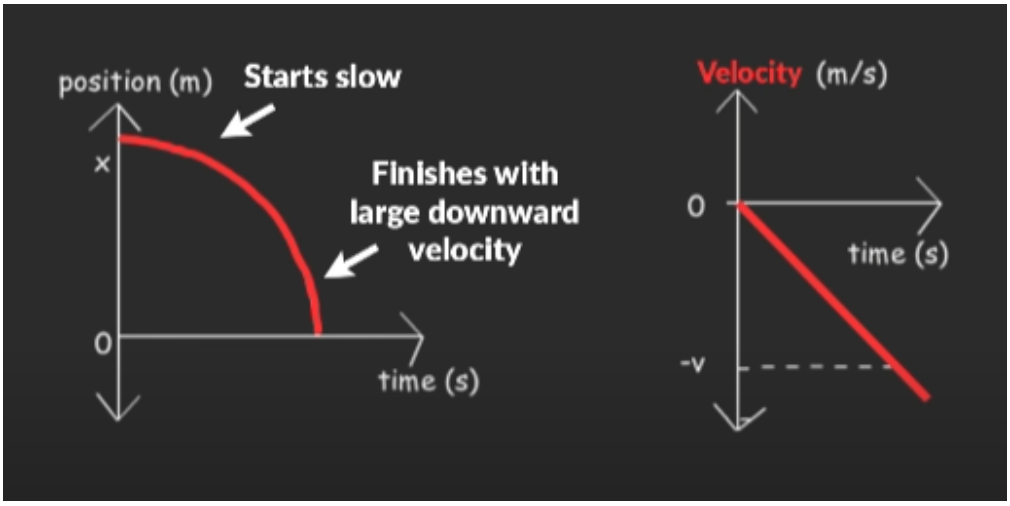


Figure 5: Explication de la force appliquée sur les joueurs

Après avoir implémenté la gravité, j'ai pu intégrer un système de saut. En appuyant simplement sur espace, notre joueur se déplace d'une certaine hauteur sur l'axe Z puis retombe jusqu'à ce qu'il rencontre le sol. De plus, j'ai fait en sorte que le joueur puisse tout de même légerement se déplacer quand il est dans les airs.

Mes prochains objectifs sont de créer des actions pour rendre le gameplay encore plus complet. Je souhaite tout d'abord permettre au joueur de s'accroupir et potentiellemnt de s'allonger, et aussi d'implémenter une touche qui permet au personnage de courrir.

### 2.3.3 Menus

Nous avons avec Paul, réfléchi et travaillé à la gestion des menus dans l'interface de notre jeu. Pour cela, nous avons tout d'abord créé un script spécialement pour gérer les menus, ce qui nous permet d'ajouter par la suite de nouveaux menus très facilement et rapidement.
Pour le moment, les seuls menus que nous avons implémentés sont ceux qui permettent aux joueurs de jouer en ligne en multijoueur, c'est à dire les menus permettant de créer une room, rejoindre une room.
Nous avons aussi créé le menu des réglages de sensibilité de la souris et de réglages graphiques, dans lequel il est possible pour le joueur de modifier l'aspect visuel de son jeu (les ombres, la qualité des textures, le niveau de détail, etc), mais aussi de changer le nombre d'images générées chaque seconde par le jeu, ou bien encore la résolution du jeu, s'il est en plein écran ou non, etc. Cela dit, nous avons déjà en ordre le fonctionnement des futurs menus que nous allons rajouter au fil de la conception du jeu, en effet en voici une représentation.

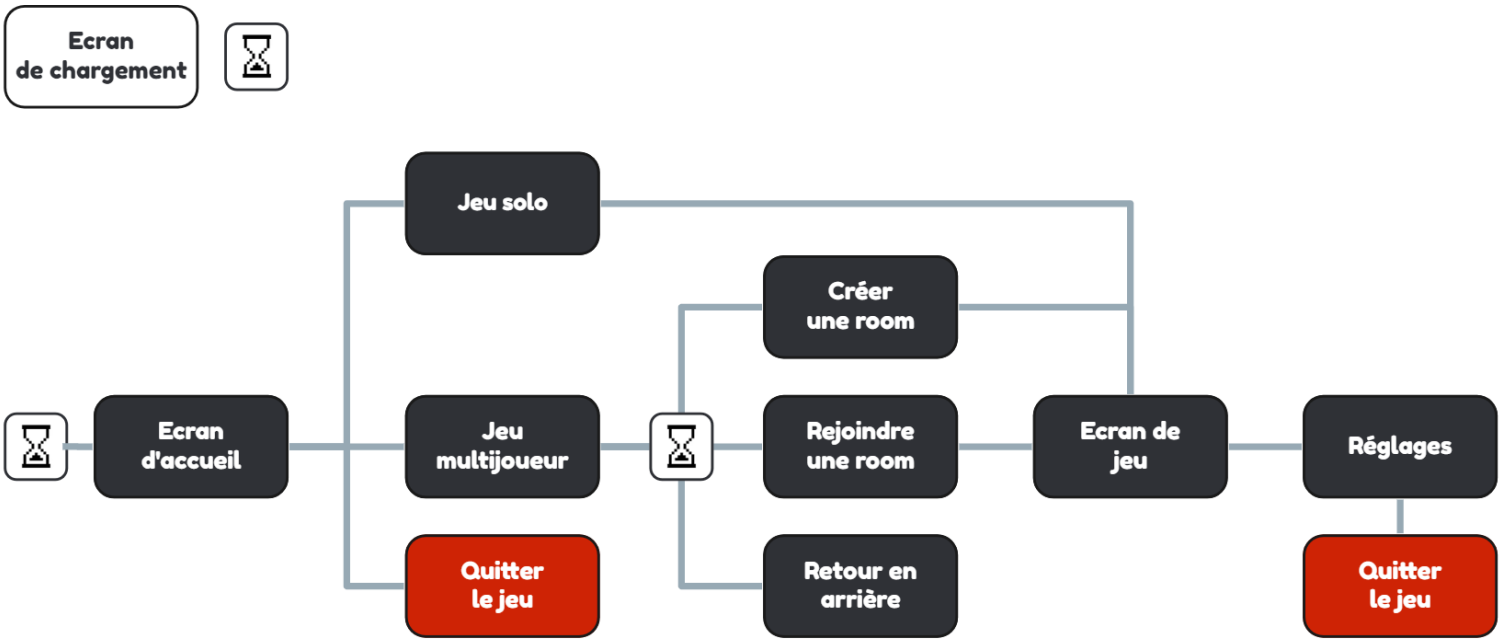


Figure 6: Architecture envisagée des menus du jeu

## 2.4 Théo

### 2.4.1 Site Web

J'ai réalisé un site web statique à l'aide de PhpStorm en HTML et CSS. Ne connaissant rien à ces langages de programmation, j'ai dû entreprendre des recherches afin d'améliorer peu à peu le site. Des améliorations pour le rendre dynamique (grâce à PHP notamment) sont en cours, afin d'implémenter un leaderboard ou un chat par exemple. Il faudrait alors envisager d'utiliser une base de donnée pour gérer les statistiques de jeu de tous les joueurs, ce qui implique d'implémenter la création de compte pour se connecter au jeu, ce qui n'est pour l'instant pas une priorité dans le développement de ce dernier.


Figure 7: Base du site web du projet

### 2.4.2 Intelligence Artificielle

Nous n'avons pas encore commencé à implémenter l'intelligence artificielle pour notre jeu car nous avons préféré avoir totalement implémenté les mécaniques fondamentales du jeu (déplacements, tirs, etc) avant de se lancer dans la création concrète de l'IA.
Néanmoins, je me suis chargé de faire de la recherche sur son fonctionnement pour être prêt et informé au moment de l'implémenter en jeu.

L'IA de notre jeu consiste à remplacer un ou plusieurs joueurs dans les mêmes parties qu'en mode multijoueur. La première chose à gérer pour cet IA est le déplacement de cette dernière. Le principe de navmesh (mesh de navigation) nous semble adapté ici, elle permettrait, accompagné d'un algorithme de pathfinding, de se diriger en direction de l'ennemi (ici le joueur, ou bien l'ennemi le plus proche). La navmesh permet de gérer en grande partie les déplacements

sur la map du jeu, en faisant éviter les obstacles de la map à l'IA, la rendant ainsi beaucoup plus fonctionnelle que si elle se dirigeait aléatoirement dans toutes les directions. Nous en implémenterons donc une au moment où nous créerons la map dans sa totalité. L'IA pourrait rechercher l'ennemi que dans certaines circonstances, si par exemple il se trouve dans un certain périmètre, etc.

Quant aux actions de l'IA, nous envisageons de créer un petit algorithme pour diriger ses actions de manière un minimum réaliste, pour la rendre plus dure à vaincre et moins mécanique, répétitive. Il faut aussi noter que nous envisageons par la suite, si le temps le permet, d'implémenter une communication entre IAs: par exemple si une partie oppose une équipe de 3 IAs et une autre équipe quelconque, nous ne voulons pas que si les 3 IAs se trouvent proches d'un ennemi, elles se dirigent toutes vers lui.
Notre idée actuelle d'algorithme est la suivante :

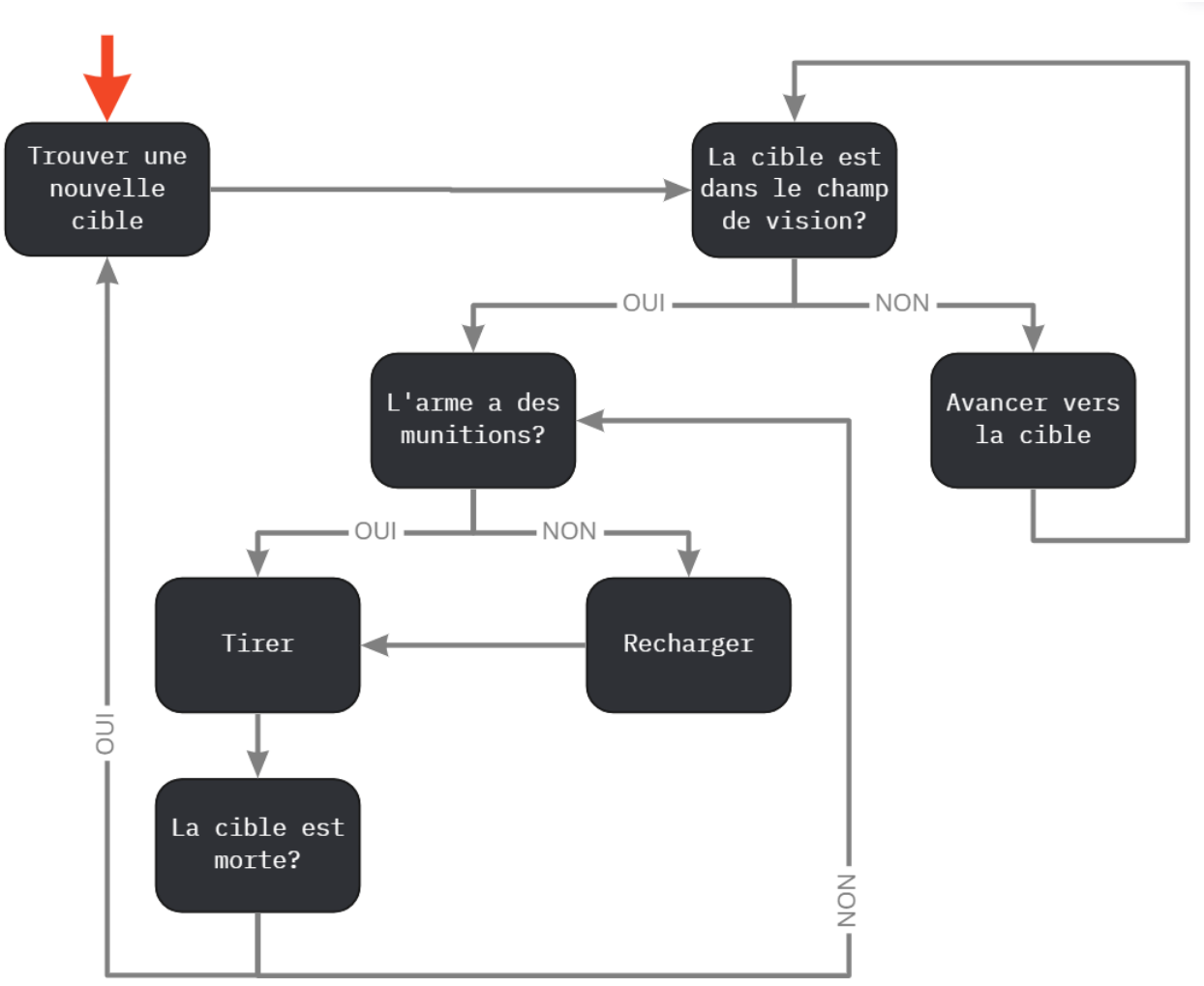


Figure 8: Base de l'algorithme envisagé pour l'IA du jeu

## 2.5 Avance/retard global

Au début de ce projet, nous avons eu du mal à instaurer des habitudes de travail régulières en raison des autres préoccupations à EPITA. Nous avons aussi rencontré plusieurs problèmes d'organisation qui ont été corrigé depuis: dans le repo git (là où sont stockés notre projet et tout nos assets), nous avions convenu d'utiliser une branche chacun dans le cadre du développement. Malheureusement, il nous est vite apparu que ce n'était pas la bonne solution tant il était compliqué de mettre nos différents travaux en commun. Nous avons donc réorganisé l'utilisation des branches git pour le développement du projet par branche de *features*, c'est à dire: créer une nouvelle branche lorsqu'on va implémenter une nouvelle fonctionnalité, pour ensuite merge avec la branche principale.

Un autre problème d'organisation a été de savoir qui fait quoi, avant quelle date. Sans document d'explication clair, il est en effet impossible de connaitre l'avancement des autres et les parties sur lesquelles ils travaillent... C'est pourquoi, comme suggéré par les ACDC, nous avons créé un document *Trello*.
Cela comporte plusieurs avantages: premièrement, il est facile de voir l'avancement général du projet et des autres membres du groupe, pour ainsi déterminer les priorités sur lesquelles se concentrer. De plus cela sert de base de ressources où l'on peut stocker tous les liens utiles et toutes les idées qu'on a eues, ainsi que les nombreux tutoriels qui ne sont pas utiles qu'à une seule personne dans le groupe.

Nous avons su nous rattraper au niveau de l'organisation du travail de groupe, et nous pensons que l'état actuel de notre jeu est en accord avec ce que nous avions prévu dans le cahier des charges.

| Tâche | Avances | Dans les temps | Retards |
|---|---|---|---|
| Graphisme | | | |
| Map | | X | |
| Armes | X | | |
| Personnages | | X | |
| Animations | | X | |
| Audio | | | |
| Musiques | | X | |
| Bruitages | | X | |
| Réseau | | | |
| Multijoueur | X | | |
| IA | | | |
| Algorithme | | X | |
| Mécanique de jeu | | | |
| Physique | | X | |
| Caméra | | X | |
| Mécanique de tir | | X | |
| Interface | | | |
| Menu | | X | |
| ATH | | | X |
| Site web | | | |
| Création du site internet | X | | |

# 3 Objectifs pour la prochaine soutenance

Maintenant que les bases de notre jeu sont bien posées, il s'agit au groupe de s'investir entièrement dans l'avancement de ce projet, en vu du travail gigantesque qu'il reste à achever. Nos objectif pour la prochaine soutenance sont les mêmes que ceux indiqués dans le tableau de planning prévisionnel, concrètement, nous souhaitons avoir un jeu jouable (c'est à dire un jeu ayant des mécaniques fonctionnelles), visuellement plus travaillé (modèles 3D des joueurs, armes et de la map, ainsi que l'interface du jeu plus accueillante), ainsi qu'une IA fonctionnelle.

## 3.1 Mécaniques de jeu

Lors de la seconde soutance, il nous sera nécessaire d'avoir totalement fini d'implémenter les mécaniques de déplacements des joueurs et de tirs pour un certain nombre d'armes, nombre que nous pourrons par la suite augmenter si la possibilité se montre. Les mécaniques du jeu ne devront plus être une priorité mais quelque chose sur lequel on peut se reposer.

## 3.2 IA

Nous allons travailler sur l'IA pour implémenter le mode solo de Nachos Rumble. En effet le mode solo repose presque totalement sur cette IA, qui rappelons le, représentera un joueur ennemi. Il est donc nécessaire que nous commençons à l'implémenter avant la seconde soutenance, pour pouvoir ensuite l'améliorer, la paufiner.

## 3.3 Graphisme

Pour la deuxième soutenance, nous prévoyons d'avoir fini la map du jeu en plus des modèles 3D d'armes que nous avons maintenant. Cela permettrait d'avoir un cadre clair aux parties du jeu. Quant au modèle du joueur, nous envisageons en avoir un, pour voir par la suite si nous en créerons plus.
Pour les animations des armes et des joueurs, nous ne nous en préoccuperons que si nous sommes en avance sur notre planning, car nous pensons qu'il s'agira plutôt de les implémenter en vue de la troisième et dernière soutenance, dans le but de finaliser et rendre beau le jeu.

## 3.4 Interface

L'interface est l'un des points où nous devons beaucoup travailler, étant donné l'état actuel de l'interface étant vraiment centré sur sa fonctionnalité et pas du tout sur l'aspect visuel du jeu, qui perd donc tout son sens.
Le schéma que l'on a réalisé pour montrer ce que l'on aimerait que nos menus deviennent va nous permettre d'être organisé dans la création et l'organisation de ceux-ci, tout en ayant les objectifs en tête.
Nous allons donc essayer d'avancer dans la création visuelle des menus finaux de notre jeu, pour rendre celui-ci plus accueillant. Il est aussi question de finaliser la création du HUD (il s'agit des informations se superposant au jeu en pleine partie sur les points de vie ou les

munitions du joueur par exemple), et d'envisager la création d'un leaderboard affichable en pleine partie, en fonction du nombre de "kills" et de morts de chaque joueurs.

# 4 Bibliographie

Voici les logiciels et outils dont nous nous sommes servis pour le développement du projet:

- Blender
- Rider
- GitKraken
- Unity
- Trello
- PHP Storm
- Photon
- Unity Asset Store
- Github

# 5 Conclusion

Nous sommes détérminés à travailler ensemble pour créer un jeu agréable et travaillé dont on puisse être fiers. Ce projet est toujours quelquechose que nous apprécions car il s'agit pour nous de l'un de nos premiers projets en autonomie complète ou presque, et surtout parce que l'idée de créer un jeu à notre image est très encourageant et motivateur.

Cette première partie du développement du projet nous a appris beaucoup de choses sur la gestion d'un travail en équipe, mais aussi dans les différentes compétences de chacun, allant de la modélisation 3D, la prise en main d'Unity et du fonctionnements de ses scripts C#, prise en main du système multijoueur de Photon, etc.

Néanmoins, le plus gros du travail n'as pas encore été fait et nous devrons continuer à nous améliorer pour pouvoir être pleinement satisfaits du résultat de notre projet.