Soutenance Finale

PastaOverflow

17 mai 2019

Table des matières

1 Introduction

Notre projet informatique de première année à l'épita est maintenant terminé. Les éléments manquant lors de la deuxième soutenance sont maintenant présents. Le jeu est maintenant arrivé dans sa phase la plus concrète et la plus finale. Il dispose donc de deux modes de jeu différents : un mode de jeu en solitaire (solo) et un mode de jeu en multi-joueur. Dans ce rapport de projet, il sera établi dans une première partie un rappel du jeu, le concept même de notre projet informatique. Nous introduirons dans une deuxième partie la façon dont nous avons conceptualisé le jeu. C'est dans une troisième partie que nous détaillerons l'état final du jeu. Dans une quatrième partie, nous aborderons la création du jeu sous l'angle des différents membres de l'équipe de développement du jeu (PastaOverflow), à travers les différentes parties contenues dans le jeu, à savoir : l'intelligence artificielle, les niveaux, les aspects sonores, les différents menus contenus dans le jeu, la modélisation en trois dimensions, l'implémentation du gameplay et le mode multijoueur. Enfin, dans une dernière partie, nous expliquerons ce que chacun des membres du groupe a gagné lors de la réalisation de notre projet informatique de fin de première année à l'épita.

2 Origine du projet et idée générale

Behind Evil Corporation est un jeu vidéo constituant notre projet au sein de notre première année à l'épita. Il est basé sur une idée de créer un jeu d'infiltration à la première personne plutôt hors du commun, la prise de risque est encouragée, contrairement aux jeux d'infiltration dans lesquels la partie se termine dès que le joueur se fait repérer (il doit intégralement recommencer la partie), dans Behind Evil Corporation, si le joueur est détecté par un ennemi, il possède encore une chance de réussite. Dans cette deuxième phase de jeu, le joueur est détecté et tous les ennemis présents dans le niveau se précipitent sur lui. Cependant, celui-ci n'est pas sans outils, il est équipé de puissantes armes permettant de se défendre contre les hordes d'ennemis l'attaquant. Pour favoriser une immersion poussée dans notre jeu, nous avons décidé qu'il serait graphiquement basé sur du graphisme type « Low Poly ». Différents détails seront à la portée du joueur. Ainsi, Behind Evil Corporation est un jeu d'infiltration à la première personne dans lequel le joueur incarne un tueur chargé d'éliminer une cible. Pour y parvenir, le joueur devra explorer les différentes cartes pour le traquer. Néanmoins, la cible possède des gardes privés, armés « jusqu'aux dents » pour protéger la cible, le joueur devra faire face. L'idée d'un tel jeu est peu ordinaire et nous a demandé un certain temps

de réflexions, par exemple, si le temps accordé par la pédagogie d'épita est suffisant pour remplir une telletâche. Nous avons choisi de prendre ce risque, prendre le risque de ne pas pouvoir terminer et pourtant, nous l'avons fait. L'idée originale du jeu était de faire un « railshooter » en réalité virtuelle. Cependant, lors d'une soirée jeux-vidéos organisée entre nous, nous avons eu l'idée de créer un jeu d'infiltration basé sur des graphismes « Low-Poly ». Notre jeu d'inspiration est « Super-Hot ».

3 Etat final

3.1 Intelligence artificielle et ennemis

Alexandre Mourère

Une de mes principales missions dans ce projet est l'intelligence artificielle. Celle-ci était prévue de la façon suivante : Les ennemis font des rondes dans les cartes, ils sont capables de détecter le joueur et de l'attaque lorsque celui-ci est repéré et à distance de feu. Dans les faits, nous avons réussi cet objectif. Nous l'avons amélioré ; les ennemis effectuent chacun leur propre ronde à travers la carte, avec un système de pathfinding amélioré. Si le joueur se fait repéré et qu'il fuir à travers la carte, les ennemis le suivent pour l'attaquer et s'il parvient à les semer, ils retournent à leurs rondes. J'ai implémenté un système de waypoints pour permettre aux ennemis de circuler dans la carte pour pouvoir faire les rondes. Le script permettant cette fonctionnalité à été totalement revue. En effet, pour permettre une meilleure fluidité des ennemis et leur permettre de tous attaquer le joueur, j'ai assemblé le script de détection des ennemis et le script leur permettant de faire les rondes et créé un script à part permettant au joueur de se déplacer. En effet, pour garantir des rondes individuelles aux ennemis (faire en sorte que chaque ennemi puisse faire sa propre ronde), j'ai utilisé des variables privées, permettant d'appliquer le script lui-même d'une façon personnalisée pour tous les ennemis. L'intelligence artificielle se décompose donc en trois scripts : le premier permettant aux ennemis de détecter le joueur et de faire des rondes, le deuxième leur permettant de « foncer » sur le joueur et le troisième leur permettant d'attaquer le joueur. Amaury Lecoq étant suppléant à l'intelligence artificielle, je lui ai permis de gérer le script permettant aux ennemis de tirer à vue sur le joueur. Je me suis concentré sur le pathfinding, la détection et les rondes des ennemis. Premièrement, concernant la détection du joueur, j'utilise un système basé sur les box-colliders de unity permettant de savoir si un GameObject (un objet physique du jeu) rentre directement en contact avec les ennemis. Nos ennemis sont dotés de deux box colliders : une physique permettant créer

l'objet physique des ennemis et une seconde, abstraite qui permet d'identifier s'il y a collision ou non avec un GameObject. A l'aide quatre fonctions, je suis en mesure de savoir si :-le joueur touche physiquement l'ennemi

— le joueur sort de la zone physique de l'ennemi
— le joueur entre dans la zone abstraite de l'ennemi
— le joueur sort de la zone abstraite de l'ennemi.

Chacun de ces cas est traité par ces quatre fonctions : si le joueur entre dans la zone physique de l'ennemi, il faut que l'ennemi arrête de courir sur lui. On utilise donc une variable privée disant à un ennemi s'il peut ou non courir sur le joueur. Si le joueur sort de la zone physique de l'ennemi, celui-ci peut donc reprendre sa course et courir sur le joueur. La variable privée utilisée précédemment doit donc permettre de reprendre sa course, d'où le choix du type de variable privée : le booléen. Ainsi, avec cette variable, nous sommes dans la possibilité de dire si l'ennemi entre en contact physique avec le joueur et s'il doit se stopper. Si le joueur entre dans la zone abstraite de l'ennemi, cela veut dire qu'il est repéré par l'ennemi en question. Il faut donc qu'il fonce sur le joueur. Cependant, il faut également qu'il prévienne les autres ennemis que le joueur est détecté. Ainsi, il faut utiliser une variable permettant de dire à tous les ennemis s'il faut tirer sur le joueur. Conséquemment, j'ai utilisé le mot clef public static permettant de rendre une variable publique et accessible dans toutes les autres classes. Un autre script est donc nécessaire pour permettre aux ennemis d'attaquer la même cible. Cette possibilité n'est envisageable que si la cible est aussi accessible dans toutes les classes. Il faut donc utiliser une variable avec le mot clef public static, représentant la cible. La cible est aussi un GameObject et, nous nommes, grâces aux fonctions précédentes, capables de détecter ce GameObject. Il m'a donc fallu modifier ces fonctions pour y ajouter la variable représentant la cible, le joueur. Donc, lorsque le joueur est repéré, détecté par un ennemi, deux variables sont mises à contribution : un booléen et un GameObject, représentant respectivement la condition « le joueur est détecté » et la cible correspondante. Nous fixons également un booléen privé à « true » (vrai) pour permettre à l'ennemi lui-même de se déplacer. Enfin, dans la quatrième fonction, nous considérons le cas dans lequel le joueur arrive à sortir de la zone abstraite de l'ennemi (ce qui revient à dire que le joueur a semé les ennemis). Finalement, grâce à ces quatre fonctions, nous pouvons indiquer à tous les ennemis s'ils doivent se déplacer vers une cible indiquée. Il reste néanmoins à implémenter les fonctionnalités permettant aux ennemis de se déplacer ainsi que de faire des rondes et de se tourner. En utilisant la documentation en ligne de unity, j'ai constaté qu'il existe une fonctionnalité très intéressante des GameObject : le Vector3.Rotate. En utilisant cette fonctionnalité, nous pouvons demander à un GameObject de se tourner vers une position indiquée. Bien évidemment,

j'ai utilisé cette fonctionnalité pour permettre aux ennemis de se tourner vers le joueur lorsque ceux-ci se déplacent vers le joueur. La question qui se pose est donc comment faire pour récupérer les postions en x, y et z du joueur pour pouvoir se tourner vers lui. Comme énoncé précédemment, le joueur possède le type GameObject, il est donc possible, via la bibliothèque unity de récupérer ces positions via des méthodes déjà existantes. Pour le déplacement des ennemis vers le joueur, j'ai, dans un premier temps, utilisé la fonctionnalité Vector3.Lerp, qui permet d'effectuer un déplacement « brute » d'un GameObject vers une certaine position, à une certaine vitesse. La position était celle du joueur lorsqu'il s'est faitrepérer par les ennemis. Cependant, même si je n'avais pas annoncé plus dans le cahier des charges, j'ai décidé d'implémenter un pathfinding des ennemis. Ainsi, depuis n'importe quel point de la carte, les ennemis sont capables d'identifier le meilleur chemin pour se déplacer jusqu'au joueur. J'ai rencontré un certain nombre de difficultés, dont celle qui faisait en sorte que les ennemis se déplacent vers la bonne position du joueur. En effet, lors de l'écriture du script, je n'avais, initialement pas pris en compte les déplacements du joueur après qu'il soit repéré par les ennemis. Ainsi, lorsque le joueur était repéré, les ennemis se dirigeaient vers la position du joueur à l'instant où il était repéré. Pour pallier cet imprévu, j'ai utilisé une fonction permettant de mettre à jour les positions en x, y et en z du joueur. Dans ma méthode update, appelée à chaque image, j'actualise la position du joueur lorsqu'il est repéré par un ennemi. Ainsi, j'ai pu implémenter la fonctionnalité permettant de faire en sorte que les ennemis puissent, théoriquement, se déplacer vers le joueur. Cependant, j'ai préféré permettre aux ennemis d'utiliser un pathfinding pour atteindre le joueur. Cette fonctionnalité sera détaillée dans une seconde partie. Maintenant que les ennemis peuvent se déplacer vers le joueur, il faut que je puisse gérer les rondes des ennemis ; chaque ennemi doit avoir son propre chemin de ronde. Aussi, il faut que tous les ennemis stoppent leur ronde lorsque le joueur est détecté et qu'ils la reprennent lorsque le joueur ne l'est plus. Ainsi, j'ai posé une condition dans ma méthode update : « Si le joueur est détecté par un ennemi, alors tous les ennemis doivent se diriger sur le joueur ». Sinon, les ennemis doivent faire leur ronde. Comment permettre aux ennemis d'effectuer leur propre système de ronde. Je me suis rappelé un attribut que possèdent tous les objets de unity : les « tags ». Ce sont des caînes de caractères permettant de connaître leur classe. Par exemple, un ennemi possède le tag « ennemi ». J'ai utilisé cette fonctionnalité pour la dériver de son but premier. J'ai créé des objets « vides » représentant les points composant la carte de ronde d'un ennemi. En effet, lors de sa ronde, l'ennemi passe par un premier point, puis par un second etc. Ces points sont les objets « vides ». Je les ai disposés sur la carte de sorte à dessiner la carte de ronde d'un premier

ennemi puis, j'ai modifié le tag des ennemis et des points pour qu'ils puissent avoir le même tag. Dans la fonction update, si le joueur n'est pas détecté, on regarde le tag du waypoint (le point correspondant aux objets vides). Si le tag du waypoint est le même que le tag de l'ennemi, alors on autorise le déplacement de l'ennemi vers le waypoint en question. Cependant, il subsiste une question : comment récupérer les waypoints déposés dans le niveau ? Pour y parvenir, j'ai placé tous les waypoints dans un autre objet vide, nommé Waypoints. Dans un autre script, je récupère tout ce que contient Waypoints et je le met dans un tableau de données appelé waypoints. Ce tableau est créé avec le mot clef public static. Il m'est donc possible d'accéder à ce tableau dans tous les scripts, dont celui permettant aux ennemis de faire des rondes. Dans ce script, nous récupérons le tableau de waypoints appelé waypoints et, avec une variable possédant le type entier, nous parcourons tous les éléments de ce tableau. L'élément courant obtenu via l'indice de lecture du tableau est appelé waypointTargeted. Nous regardons en premier lieu s'il n'est pas nul, auquel cas nous ne faisons rien. Sinon, nous comparons le tag de waypoint-Targeted et celui de l'ennemi en question. S'ils sont différents, nous modifions l'indice de lecture de sorte à ce que si nous regardions le dernier élément de waypoints (ce qui revient à comparer le nombre d'éléments de waypoints et l'indice de lecture), nous décrémentons l'indice de lecture : l'ennemi va faire sa ronde à l'envers, il revient, dewaypoint en waypoints jusqu'à son premier waypoint. Si nous regardions le premier élément de waypoints ou si le tag de waypointTargeted est différent du tag de l'ennemi en question, nous incrémentons l'indexe de lecture. Ainsi, nous regardons le waypoint suivant. Si le tag de waypointTargeted est le même que le tag de l'ennemi, alors nous utilisons la fonctionnalité Pathfinding pour rejoindre le waypointTargeted. Nous créons un NavMeshAgent agent qui correspond au NavMeshAgent de unity, une fonctionnalité permettant d'implémenter le pathfinding dans unity. Ce NavMeshAgent agent est initialisé à partir de l'ennemi actuel. C'est avec la méthode SetDestination que nous demandons à l'agent correspondant à l'ennemi actuel de faire un déplacement, avec pathfinding, ves la postion de waypointTargeted. Une fois cette fonctionnalité implémentée, les ennemis se dirigeaient vers les waypoints, avec chaque ennemi ayant sa propre carte de rondes. Cependant, les ennemis n'arrivaient pas à atteindre la position exacte du waypoint : les ennemis essayaient d'arriver sur la postion exacte du waypoint, ce qui est pratiquement impossible, ils étaient très proche de leur waypoint mais pas assez pour être exactement dessus. Ainsi, j'ai utilisé la méthode Distance permettant de connaître la distance entre deux positions : je regarde si la position (en Vector3) de l'ennemi moins la position (en Vector3) de waypointTargeted est inférieure ou égale à 0.5. Ainsi, si l'ennemi est suffisamment proche du waypoint, il pase au suivant. Finalement,

j'ai donc réussi à implémenter les rondes sur les ennemis du jeu, avec chaque ennemi ayant sa propre carte de rondes et un pathfinding permettant d'accéder à tous les waypoints, lorsque le joueur n'est pas détecté par les ennemis. Maintenant, nous allons voir comment il m'a été possible d'implémenter le pathfinding sur les ennemis afin qu'ils rejoignent le joueur. Rappelons-nous qu'avec les quatre fonctions décrites précédemment, nous renvoyons la position du joueur, ainsi qu'un booléen public permettant de dire que le joueur a été détecté. Dans un nouveau script, nous récupérons ces variables pour savoir si les ennemis doivent attaquer le joueur. Si c'est le cas nous créons, comme pour le système de rondes, un nouvel agent NavMeshAgent et nous le faisons se déplacer vers la position du joueur à l'aide d'un algorithme de pathfinding. Une question importante peut-être soulevée : pourquoi avoir créé un nouveau script ? La réponse est parce que les ennemis doivent attaquer la même personne : la cible est le joueur. Si toutes ces actions étaient réalisées dans un seul script, sans variables publiques, seul l'ennemi ayant détecté le joueur pourrait se déplacer vers lui. Ainsi, il nous fallait utiliser deux scripts distincts.

Enfin, lors de notre cahier des charges, nous avons annoncé que nous voulions créer trois types d'ennemis différents. Cette tâche étant dédiée à la section intelligence artificielle, je l'ai donc implémentée. En effet, nous avions prévu un ennemi de type « light » (léger en anglais) ; un autre de type « medium » (moyen en anglais) et enfin un ennemi de type « heavy » (lourd en anglais). Nous avons également précisé les différentes caractéristiques de chaque ennemi. Le premier étant le light (léger), nous lui avons donné une vitesse importante, des points de vie réduits et des dégâts faibles. En effet, nous considérons cet ennemi comme un éclaireur : il se déplace plus rapidement, fait moins de dégâts et possède des points de vie plus limités.

Ensuite, vient le second type d'ennemis : le « medium » (lemoyen). Celui-ci correspond à l'ennemi de base, il s'agit de l'ennemi le plus présent sur les cartes. L'ensemble des caractéristiques de ce type d'ennemi est assimilable à celui d'un ennemi normal, ses caractéristiques sont basiques : des points de vie équilibrés sur la vitesse et les dégâts. Le calcul des points de vie, de la vitesse et des dégâts est arbitraire. J'ai donc effectué des choix de gameplay pour le jeu. Enfin, le dernier type d'ennemi est celui dit « heavy » (lourd en anglais). L'ennemi possédant ce type sera capable d'infliger un certain nombre de dégâts, plus que l'ennemi léger et moyen, sera plus lent que l'ennemi moyen (et donc bien plus lent que l'ennemi léger). Ses points de vie seront les plus élevés des autres types d'ennemis. De plus, étant donné que je suis également responsable de la section niveaux, j'ai pu appréhender les besoins que nécessitent les niveaux. Ainsi, la cohésion du développement du jeu est permise par l'attribution des différentes tâches de chaque membre du

groupe, effectué lors du cahier des charges. Pour y parvenir, j'ai décidé de baser l'ensemble des types d'ennemis sur un seul type, modifié pour chaque type. Les caractéristiques de chaque ennemi étant choisies arbitrairement, j'ai pu parvenir à équilibrer l'ensemble des caractéristiques des ennemis pour éviter qu'un type d'ennemi soient trop puissant ou trop faible par rapport aux autres. Les décisions concernant ce choix ont été prises en prenant en compte le gameplay du joueur et les niveaux. Finalement, les ennemis peuvent attaquer le joueur, tout en étant équilibrés pour ne pas freiner la progression du joueur et ainsi favoriser le gameplay et optimiser l'expérience de jeu. En bonus, j'ai décidé d'ajouter un dernier type d'ennemi, celui du boss de fin du jeu. Cependant, étant donné que nous n'avions pas indiqué que nous le ferions, ce dernier type d'ennemi est considéré comme un des nombreux bonus présents dans le jeu. Ainsi, dans le domaine de l'intelligence artificielle, tout ce qui a été prévu a été fait. De plus, j'ai ajouté les bonus détaillés ci-dessus. Finalement, j'ai donc implémenté une intelligence artificielle perfectionnée de sorte qu'elle puisse parcourir la carte en faisant des rondes, avec pathfinding, modifiables par les responsables de la section Niveaux, détecter le joueur à distance et le rejoindre à l'aide d'un algorithme poussé de pathfinding. Aussi, il existe 3 types d'ennemis ayant chacun leurs propres spécifications. Ainsi, l'intelligence artificielle est complètement fonctionnelle, adaptée à un jeu d'infiltration. En effet, les ennemis (déclinés en trois types différents et possédant chacun leurs propres caractéristiques) sont désormais capables de faire des rondes, d'attaquer à distance le joueur lorsque celui-ci est détecté par au moins un ennemi et de se déplacer vers lui en utilisant un algorithme amélioré de pathfinding. C'est donc pourquoi les missions qui m'ont été confiées dans ce domaine ont donc toutes été réussites. En tant que responsable général de l'intelligence artificielle du projet, j'ai implémenté les ennemis sur toutes les cartes et ait expliqué aux autres membres du groupe la façon dont l'intelligence artificielle fonctionne pour leur permettre d'implémenter également des ennemis sur les cartes. Ainsi, j'ai dû expérimenter un travail de communication au sein même du groupe pour informer l'ensemble des membres. Mon travail dans l'intelligence artificielle, bien que complexe, ne s'est pas déroulé sans difficultés, voire des difficultés inattendues. En effet, les complications rencontrées ont été, et dans le domaine propre de l'intelligence artificielle : comment faire en sorte que toutes les fonctionnalités de l'intelligence artificielle des ennemis puissent s'emboîter parfaitement. Ce type de difficulté sera traité dans la section concernant les complications que nous avons rencontrées lors dela création de notre projet informatique. De plus, une nouvelle difficulté est apparue dans mon travail : les techniques utilisées lors de la création de l'intelligence artificielle des ennemis sont entrées en conflit avec d'autres techniques utilisées par mes collègues, notamment dans

le domaine correspondant à la capacité du joueur à tirer, à toucher et à tuer les ennemis. Nous traiterons également cette difficulté plus en profondeur dans la section dédiée. Ainsi, l'intelligence artificielle a été un véritable succès et correspond à un atout majeur de notre projet informatique de fin de première année au sein de l'EPITA.

3.2 Niveaux

Alexandre Mourère

Une autre de mes missions au sein de ce projet informatique de fin d'année était l'élaboration des niveaux. Cette partie, contrairement à la section intelligence artificielle, nécessite moins de capacités de programmation. Cependant, j'ai tout de même implémenté quelques scripts permettant l'amélioration des niveaux et du gameplay. Ainsi, les niveaux seront plus fluides et dynamiques, néanmoins nous reviendrons sur cet aspect de développement dans une partie prévue à cet effet. Concernant la section dédiée à la construction des éléments virtuels représentant des éléments physique dans le jeu, appelée map ou carte ou bien encore level en anglais, j'ai eu pour mission de conceptualiser, inventer, implémenter et texturer 3 cartes. Cette mission a été terminée, l'objectif des trois missions est atteint. En effet, lors de notre cahier des charges, nous avons indiqué notre volonté d'implémenter deux mondes de trois cartes. Un monde, dans le langage du level design, correspond à un univers, il possède des attributs qui lui sont propres, tels que l'ambiance, le nombre de cartes ainsi que le type d'ennemis. Ainsi, nous avons décidé d'implémenter deux mondes. Les cartes correspondent à des niveaux dans le jargon du level design. Un niveau est un ensemble d'éléments cohérents, on note qu'un synonyme deniveaux dans le langage des jeux vidéo map. Lors de la conception de mes niveaux, j'ai dans un premier niveau décidé d'utiliser notre principe du jeu :

Le premier niveau possède deux chemins : un premier chemin permettant au joueur de s'infiltrer silencieusement à travers la carte. En effet, si le joueur décide au début de tourner à gauche, il devra essayer d'éviter de se faire repérer dans les couloirs par les ennemis. Ce chemin illustre parfaitement l'esprit premier du jeu : il s'agit d'un jeu d'infiltration dans lequel le joueur devra faire face à deux ennemis légers (des éclaireurs). Pour progresser dans le couloir, il devra donc les supprimer sans se faire repérer, sous peine de devoir affronter de face tous les ennemis. Lors de sa traversée, de son infiltration, le joueur appréciera les nombreux détails présents sur la carte. À savoir des machines industrielles, des caisses en bois et des cylindres. Tous ces éléments de décors sont bien évidemment détaillés, avec la plus grande des intentions.

En effet, par exemple, les machines industrielles sont de couleur jaune et possèdent un niveau de détail impressionnant. Ce qui permet au joueur de se projeter au sein du jeu, d'une façon bien plus réaliste. L'optimisation de l'expérience de jeu se fait également à travers la conception et l'implémentation des niveaux. Ces machines industrielles s'intègrent parfaitement dans le décor, dans l'environnement de la carte dans laquelle il est possible de la trouver. L'ambiance industrielle du lieu est propice aux éléments de décors tels que les machines industrielles. D'un point de vue plus technique, ces machines servent à assembler des éléments tels que des voitures, des téléphones ou bien des télésièges. Le joueur peut donc plonger dans une usine et découvrir un univers industriel dans lequel il n'avait pas encore découvert les plus profonds des détails. Pour trouver cette machine industrielle, j'ai parcouru les eaux profondes de l'internet. En effet, certains rares éléments de décors peuvent être mis à disposition de certains créateurs afin de les aider à embellir leurs créations. Ces éléments m'ont été accessibles dans la mesure où, après avoir effectué des recherches, j'ai trouvé ces éléments de décor. L'endroit dans lequel il m'a été possible de trouver des éléments de décors possédant un tel niveau de détail est appelé « Asset Store ». Les créateurs les plus performants peuvent y trouver des éléments impressionnants. C'est donc ainsi que j'ai pu découvrir ces machines industrielles. Néanmoins, ces machines ne sont que des diamants bruts auxquelles il faut, à l'aide d'un travail conséquent, ajouter finement des éléments permettant d'obtenir des créations surprenantes. L'élément dont il est ici question est l'emplacement dans la carte. Effectivement, il faut placer, avec une certaine précision, ces éléments pour leur donner une signification pertinente. Il m'a semblé judicieux de les placer à l'entrée, près des caisses en bois (elles-mêmes modélisées et texturées par nos soins). Ces caisses en bois sont le fruit d'une collaboration effectuée entre trois membres du groupe PASTAOVERFLOW (Jean- Philippe BINGER, Benoît MALHOMME et moi-même, Alexandre MOURERE). Benoît MALHOMME a créé une texture permettant de ressembler à du carton, tandis que Jean- Philippe BINGER modélisait lesdites caisses à l'aide de logiciels informatiques complexes de modélisation en trois dimensions. Enfin, Alexandre MOURERE les ait déplacés afin d'obtenir en emplacement pertinent dans ma carte. Ainsi, il m'a été possible d'implémenter des machines industrielles à des endroits stratégiques sur la carte pour permettre de favoriser une expérience de jeu optimale pour le joueur. De plus, les nombreux réservoirs présents sur la carte sont de couleur verte et possèdent des textures qui permettent au joueur uneimmersion totale dans le jeu. Ce « feature » (contenu en anglais) fait parti d'une famille d'objets utilisée pour enrichir la carte en question. En plus des machines industrielles, le joueur doit avoir l'illusion d'être lui-même dans une usine. Cela est possible grâce à l'ajout d'éléments décoratifs tels

que des pilonnes. De la même manière que pour les machines industrielles, les pilonnes ont été trouvés, dans un état brut, sans effet additionnel. C'est ici qu'intervient une partie de mon travail : arriver à coordonner un ensemble d'éléments décoratifs pour permettre au joueur une immersion dans notre jeu, et plus particulièrement dans l'univers industriel. Lorsque le joueur traverse le premier couloir, il accède à un autre couloir dans lequel se situent deux ennemis. Ceux-ci effectuent des rondes dans la carte. Le premier est très proche du joueur. En effet, pour ne pas se faire repérer par cet ennemi, le joueur devra être rapide et furtif. Dès que l'ennemi en question a le dos tourné, le joueur doit s'engouffrer dans le conteneur ouvert. Ce conteneur est détaillé de telle sorte à ce qu'il fasse partie intégrante du la carte. Il permet de préserver l'ambiance industrielle de la carte, tout en assurant de ne pas troubler la cohérence des éléments décoratifs énoncés précédemment. Cependant, lorsque le joueur traverse ce conteneur, il doit faire vite : le deuxième ennemi évoqué ci-dessus effectue ses rondes, il faut donc qu'il se dépêche d'avancer et de se cacher ailleurs, le temps que cet ennemi parte recommencer sa ronde. Le joueur pourra apprécier les détails présents à cet endroit de la carte, où les conteneurs et autres machines industrielles lui seront utiles pour pouvoir se cacher de l'ennemi. Les détails installés par mes soins sont, par exemple, des machines industrielles, des conteneurs texturés, ainsi que des caisses en carton. Celles-ci servent à stocker des objets et s'intègrent donc bien dans l'ensemble du niveau, de la carte. Enfin, après avoir soit éliminé soit esquivé un dernier ennemi, l'ennemi doit courir jusqu'à la sortie du niveau, caractérisée par une porte verte. Nous avons choisi, moi ainsi que l'autre responsable de la section « Niveau » ce code couleur : -porte verte correspond à la sortie du niveau : le joueur doit chercher cette porte pour terminer le niveau. Le vert a été choisi parce qu'il incarne l'aspect positif, c'est un choix de game designer que nous avons décidé de faire.

Une porte rouge correspond à l'entrée du niveau : celle-ci demeure fermée, le joueur ne peut pas revenir dans un niveau déjà terminé. Ce choix a été fait dans le même temps que celui du code couleur « porte verte pour sortie ». Il correspond au choix logique et habituel des jeux vidéo : si l'esprit de notre jeu dépayse totalement le joueur (un jeu d'infiltration dans lequel il est possible de passer vers un mode plus nerveux lorsque le joueur est repéré), nous avons cependant décidé de conserver les codes de jeux actuellement en vigueur. Ces codes de jeux sont des consensus dans lequel le vert désigne le bon tandis que le rouge désigne le mal. Il existe donc un lien puissant entre les derniers jeux actuels et le nôtre. Ainsi, si le joueur décide de tourner à gauche dès le début du jeu, il doit s'infiltrer parmi les différents éléments décoratifs en place pour pouvoir accéder à la sortie. Nous allons à présent étudier ce que le joueur rencontre lorsqu'il décide de tourner à droite. Si cela

correspond au choix du joueur, celui-ci rencontrera dans un premier temps un outil très utilisé dans le milieu industriel : un monte-charge. Ceci est une machine rappelant l'esprit industriel de la carte, et du monde. Par monde, nous entendons ici l'ensemble des cartes ayant le type industriel. Ce monte-charge, tout comme les machines industrielles proviennent de l'Asset Store de unity.

Initialement, il s'agissait d'un diamant brut. Néanmoins, avec le travail fourni, nous pouvons constater à quel point il s'intègre parfaitement dans le décor et même plus, en contribuant à l'aspect esthétique du jeu. C'est donc un véritable atout pour cette carte, présent à trois endroits différents. Le joueur pourra également observer la présence d'un bureau, modélisé et texturé par mes soins à l'aide des outils présents dans unity. En effet, ce bureau d'apparence brun simule le plan de travail, le bureau du service de sécurité de l'usine. Ainsi, un ordinateur portable, texturé est placé sur le bureau. On notera également la présence d'une chaise de bureau à roulette bleue en face du bureau pour indiquer au joueur que l'usine n'est pas abandonnée. Cependant, on notera la présence d'un rendu vidéo sur un grand écran sur la droite. En effet, j'ai décidé d'implémenter une caméra de sécurité, qui détecte le joueur et qui retransmet, en temps réel ce qu'elle voit. Pour y parvenir, j'ai été contraint d'implémenter un script permettant de récupérer le rendu vidéo d'une caméra de sécurité. Ces caméras sont au nombre de deux dans toute la carte. Si le joueur passe devant l'une de ces caméras il se fera repérer et automatiquement, tous les ennemis se dirigeront vers lui dans le but de l'attaquer. Pour parvenir à un tel résultat, j'ai divisé le problème en trois sous-problèmes : le premier est de choisir, en tant que game-designer où placer les caméras et où placer le rendu vidéo de celles-ci. J'ai trouvé judicieux de placer les deux caméras à la fin du niveau de sorte que le premier chemin, celui dans lequel le joueur doit s'infiltrer, reste équilibré et encourage toujours le joueur à essayer le côté infiltration de notre jeu. Il m'est également apparu qu'il est inutile de placer des caméras de sécurité dans le chemin où le joueur se fera nécessairement repérer par les ennemis. Ainsi, j'ai choisi de placer les caméras de sécurité en fin de niveau. De cette façon le joueur devra, s'il choisit l'infiltration, s'empresser d'atteindre la sortie du niveau pour ne pas mourir. Si le joueur décide d'attaquer de face les ennemis (en prenant à droite au début du niveau), et s'il arrive à éliminer une partie des ennemis mais parvient à semer les autres, celui-ci se fera nécessairement repérer par les caméras de sécurités présentes dans la carte. Ainsi, l'emplacement des caméras de sécurité (rigoureusement choisit) permet au joueur d'avoir un cran de difficulté plus élevée, il sera donc maintenu en haleine et essayera donc de se performer au jeu. Le deuxième sous- problème résidant dans l'implémentation des caméras de sécurité est de parvenir à dé-

tecter le joueur à partir d'une caméra de sécurité. En effet, si les caméras de sécurité ne permettaient pas la détection du joueur, celles-ci ne seraient d'aucune utilité ! Pour pouvoir détecter le joueur, j'ai donc choisi un ennemi, lui ait ôté des scripts de sorte qu'il ne puisse pas se déplacer (qu'il soit immobile donc). Sauf le script permettant la détection du joueur. Comme la zone de détection des ennemis est modulable, je l'ai modifiée afin d'obtenir une surface de détection permettant d'alerter tous les gardes aux alentours. La question est donc comment faire une telle chose. J'ai trouvé la réponse dans les options des zones de collision des ennemis. En effet, pour la détection des ennemis, j'utilise une zone de détection physique et une zone invisible (cf partie correspondante dans la section intelligence artificielle). J'ai modifié la zone abstraite des ennemis, à l'aide d'outils relativement complexes mais intégrés à unity. Ces outils permettent, en sélectionnant le bon composant de modifier les variables publiques présentes dans les scripts par exemple. Je me suis donc appuyé sur cet outil pour arranger la zone de détection pour qu'elle puisse être assez grande afin de détecter le joueur sur une certaine zone. Puisque la mise en place de caméras de sécurité nécessite un ennemi, dont la zone de détection est modifiée, il étaitdonc obligatoire de placer ledit ennemi quelque part sur la carte. Cependant, puisque j'ai retiré tous les composants liés à cet ennemi, celui-ci ne peut donc pas se déplacer ni tirer sur le joueur. J'ai donc, dans un premier temps retirer son « skin », son aspect pour qu'il apparaisse invisible aux yeux du joueur. Ainsi, le joueur ne peut pas voir l'ennemi mais peut l'éliminer. Ainsi, dans un second temps, j'ai donc déplacé cet ennemi dans un endroit inaccessible pour le joueur : sous la carte. C'est donc de cette façon que j'ai implémenté un ennemi invisible, réduit à un script « vivant ». Le script en question est bien entendu le script permettant aux ennemis de détecter le joueur. Ainsi, lorsque le joueur se fait repérer par cet « ennemi », le script est appelé, les ennemis aux environs de la carte sont alertés et se dirigent vers le joueur. Il est donc possible d'ajuster les paramètres pour faire en sorte de modifier la zone de détection de la caméra de sécurité permettant ainsi d'augmenter le réalisme de notre jeu vidéo. Ainsi, les compétences acquises lors de conception et lors de la création de l'intelligence artificielle des ennemis se sont avérées indispensables : sans les connaissances, il m'aurait été impossible de penser à modifier un ennemi, en lui retirant tous ses composants, pour le réduire à un seul script et à une zone de détection invisible. Le fait d'être en même temps responsable de l'élaboration des niveaux et de l'intelligence artificielle des ennemis m'a donc permis de développer des caractéristiques et de nouveaux objets et concepts qui auraient été impossibles sans la dualité niveau/intelligence artificielle des ennemis. Le dernier sous-problème est celui correspondant au rendu, en temps réel, de ce que « voient » les caméras de sécurité. J'ai donc

ajouté un script, permettant d'assimiler un objet du jeu (un GameObject dans unity) à un poste de rendu vidéo. Le script en question fait partie intégrante du paquet (« package » dans unity) dans lequel se trouve le corps physique de la caméra. Encore une fois, la caméra seule ne « vaut rien », pour lui en donner, il a tout d'abord fallu la placer rigoureusement dans la carte et ensuite lui permettre de détecter le joueur et enfin d'afficher son rendu vidéo en temps réel dans le jeu. Ceci est possible grâce au script fourni dans le package de la caméra. Il faut néanmoins relier le GameObject de rendu vidéo en temps réel à la caméra. Une difficulté m'est soudainement apparue : quel objet choisir pour pouvoir afficher, en temps réel, le rendu de ce que voient les caméras de sécurité ? La réponse fut pour moi une évidence : un écran. Lors du développement des niveaux, j'ai obtenu un écran, permettant de montrer au joueur que des salariés travaillent dans la carte. Cet écran possède une taille prédéfinie. Néanmoins, à l'aide des outils intégrés à unity, il m'a été possible de redimensionner l'écran d'ordinateur pour en faire un plus grand, possédant une surface d'affichage plus grande et lui donner un aspect de grandeur qu'il ne possédait pas auparavant. L'écran sensé pouvoir retranscrire le rendu vidéo en temps réel de la caméra est donc terminé : il ne reste plus qu'à assembler toutes les pièces du puzzle pour pouvoir « donner vie » à mes caméras de sécurité. J'ai donc sélectionné l'écran nouvellement formé dans les composants de la « scène » (la carte en développement) pour qu'il puisse recevoir le rendu vidéo en temps réel de la caméra. Aussi, j'ai placé soigneusement l'ennemi réduit à un script et une zone de détection. Un dernier problème est apparu : la caméra ne tournait pas. Pour la forcer à tourner, j'ai utilisé le script fourni pour modifier la variable représentant la vitesse de rotation de la caméra. Cependant, le rendu-vidéo était décalé : l'affichage était en mode portrait et avait une rotation d'angle de 90 degrés. J'ai donc modifié les valeurs du GameObject représentant les coordonnées en jeu de l'écran de surveillance afin de permettre de modifier la rotation même du rendu vidéo entemps réel de la caméra de surveillance, en appliquant une rotation de 90 degrés à l'écran. Le problème de la rotation d'angle et du mode portrait était donc réglé. Un problème persistait néanmoins : celui du mouvement de la caméra. Celui-ci ne correspondait pas aux rotations effectuées par la caméra de sécurité. Pour pallier cette difficulté, j'ai modifié la rotation initiale de la caméra, en anticipant mentalement la rotation de celle-ci une fois en jeu, pour donner une meilleure fluidité au rendu vidéo en direct de la caméra de sécurité en question. Ainsi, les caméras de sécurité détectent le joueur et alertent tous les ennemis présents sur la carte lorsque le joueur est effectivement détecté. Quant à l'emplacement de l'écran retransmettant en temps réel le rendu d'une des deux caméras de sécurité, j'ai décidé de la placer au début du niveau. En faisant ce choix, j'avais pour idée de permettre au

joueur d'utiliser ces caméras de sécurité pour pouvoir lui permettre d'identifier tous les ennemis présents en fin de niveau, pour l'aider à établir une stratégie d'attaque. Les caméras de sécurité sont donc en plus, en élément de gameplay très intéressant, permis par le fait de mon travail en tant que responsable de l'intelligence artificielle des ennemis et responsable des niveaux. Le joueur peut donc accéder, dès le début du niveau au nombre d'ennemis présent en fin de niveau et peut estimer les déplacements des ennemis. En effet, puisque les ennemis effectuent des rondes dans la carte, anticiper leur déplacement est chose aisée lorsqu'il est possible de connaître les points de déplacements ennemis. Ainsi, le joueur pourra élaborer sa propre stratégie pour franchir la carte et accéder à la fin du niveau. En implémentant ces caméras, j'ai donc rendu chaque partie dans ce niveau unique.

Le deuxième niveau se situe dans le deuxième monde : celui des bureaux. Dans ce monde, j'ai choisi d'implémenter dans un premier niveau une sorte de labyrinthe dans lequel se situent des ennemis, faisant bien évidemment des rondes. En tant que responsable de la section Niveaux, j'ai décidé de poser des bureaux équipés d'ordinateurs, de plantes, d'écrans ainsi que des détails tels qu'un parapluie posé sur un des bureaux pour proposer au joueur un réalisme plus poussé, qui lui permettra de mieux s'immerger dans mon niveau. Dès le début du niveau, le joueur se fera repérer par un ennemi. Le joueur est donc confronté à un choix : soit tuer tous les ennemis sans discrétion ; soit fuir cet ennemi pour aller se cacher et passer dans la phase d'infiltration du jeu. Ainsi, le principe du jeu est donc respecté, une nouvelle fois, dans mes niveaux. Lors de l'infiltration, le joueur devra explorer chacune des pièces se trouvant sur la carte, il trouvera de ce fait des salles dédiées aux réunions. Celles-ci sont équipées de grandes tables, d'écran d'ordinateur, de grands écrans ainsi que des ordinateurs portables. De plus, j'ai ajouté des chaises autour de ces tables afin de montrer au joueur que ces bureaux sont normalement habités par des salariés. Une fois le dernier ennemi abattu ou une fois que le joueur parvient à accéder à la fin du niveau, il retrouvera la porte verte correspondant à la sortie de la carte.

Le troisième et dernier niveau se situe lui aussi dans des bureaux. Cependant, ce niveau contraste avec les précédents dans la mesure où le joueur sera détecté dès le début du jeu par les ennemis. Dans le cahier des charges, j'avais indiqué un niveau dans lequel le joueur devra obligatoirement « foncer dans le tas ». C'est de ce niveau qu'il s'agit. Le joueur sera confronté à des ennemis très nombreux et armés jusqu'aux dents. Néanmoins, des armes sont à la disposition du joueur dès le début. Ainsi, celui-ci pourra faire face à la horde d'ennemis. Au début, le joueur aura l'impression d'être enfermé : il n'y a pas de porte pours'échapper. En effet, j'ai implémenté un script permettant de détecter le nombre d'ennemis en vie. Si les tous les ennemis sont morts, j'ai

décidé de détruire une porte, afin de permettre au joueur de s'échapper. La fin de la carte n'est donc accessible qu'après avoir éliminé tous les ennemis. Cette fin correspond à un garage, comportant un certain nombre de voitures garées. Pour quitter le niveau, le joueur devra monter dans un taxi, ce qui diffère de l'habitude créée jusqu'ici.

Jean-Phillipe BINGER

Les niveaux ont été implémentés en deux parties, une partie sur le thème industriel et la seconde partie est sur le thème des bureaux.

Les niveaux industriels :

Les niveaux industriels sont de grande entre coupé de petites carte permettant la transition entre deux grandes cartes chacune des cartes comporte ces propositions de pour le joueur choix , son nombre d'ennemis et leur ronde et bien chacune à son propre design lui permettant de ce différencié des autres carte et de varier l'expérience visuel du joueur.

La première carte que j'ai fais et une carte représentant un hangar où sont entreposé des caisses. Toutes les caisse, tous les barils et tous les conteneurs ont été placés de manière à rende la carte la moins répétitive possible faisant même passer le joueur sur une passerelle au dessus de l'ensemble du hangar. Les ennemis sont disposé tout le long du chemin faisant ainsi des rondes sur ce dernier si le joueur souhaite passer discrètement ce dernier doit avoir de très bon timing afin d'évitée le combat. La sortie du niveau et représenté par une porte qui est une sortie de secours si le joueur s'approche de cette dernière il passe alors au niveau suivant en passant d'abord par une petite carte qui permettant de faire le passage d'un niveau à l'autre.

La seconde carte que j'ai faite est la troisième carte industrielle cette carte et une carte ou le joueur à plus de liberté de mouvement lui laissant la possibilité de passé entre des grosse ligne de machines ces dernière étant à l'arrêt le joueur ne peux donc pas compter sur le bruit que ces machines produirais pour passer discrètement il lui faudra repérer ces ennemis de manière distante pour pouvoir passer sans se faire voir ou bien le joueur peux choisir de jouer de manière plus agressive en éliminant tous les ennemis se trouvant sur son chemin. Les ennemis encore une fois sont postés à des endroits précis pour que joueur puisse les repérer si ce dernier ne se précipite pas. Des armes sont mises a disposition du joueur afin qu'il puisse se défendre si il venais à se faire repérer ou bien à directement attaquer ces adversaires. En effet différent détail sont apporté par l'ajout de caisse étant le produit du travaille de plusieurs membres du groupe dont Benoit MALHOMME qui a permit d'intégrer les caisse dans le jeu de manière texturé.

Mon rôle a aussi été de mettre en place les texture principale sur les cartes

industriels mettant en place ainsi un visuel plus attractif pour le joueur que simple objet ne possédant qu'une seule couleur pour se faire j'ai dû utiliser le système de matériaux proposé par Unity ces derniers permettent de donner à l'entièreté d'une surface une texture ainsi qu'un certain nombre d'aspect. Les textures et les aspect utilisé on été fourni par Benoit MALHOMME qui à travailler ces textures afin qu'elle puissent s'appliquer au mieux à l'environnement de Unity cependant un problème a été rencontrer lors de la mise en place des matériaux sur les objets car si l'on voulait deux objet avec une même texture et un même aspect mais avec une répétition différente il fallait alors créer un nouveau matériel car si l'on modifié la répétition sur un objet celle-ci était changé pour le matériel en lui même conduisant à une modification sur tout les objets utilisant ce matériel. Pour solutionner ce problème nous avons créer différents matériaux mais nous avons aussi limité leurs nombres en faisant en sorte que les objet utilisant la même texture puisse avoir la même répétition en utilisant des tailles proches rendant les changement de manière à ce qu'il soit imperceptible.

L'ajout de texture a servie en général à rendre le joueur plus immergé dans sa partie puisque se dernier va voir les différentes textures sur les objets, mur, sol et plafond de tel manière à ce que cela lui fasse pensé directement au thème de l'industriel puisque le sol est texturé de béton, les murs quand à eux ont une texture faisant pensé à du métal ondulé que l'on retrouve souvent au niveaux des hangars de stockage industriel mais pas seulement l'ajout des textures sur les différents objet l'environnant auront le même effet.

Je me suis aussi occupé de créer une carte pour le multijoueur cette dernière doit être suffisamment grande et équilibré pour que plusieurs joueur puissent en même temps combattre sans être proche les uns des autres ainsi une forme hexagonale semble tout bonnement être plutôt équilibré puisque ceci empêche les joueur de bloquer tout les secteurs de la carte. Il m'a fallu pour cette carte ajouté des systèmes de «spawnpoint» qui sont les points d'apparition que ce soit pour les joueur ou pour les ennemis et ainsi aussi empêcher les ennemis d'apparaitre directement sur le joueur, cette carte comporte une bonne quantité d'obstacle permettant aux joueurs de ce mettre à couvert mais ne lui laissant pas la possibilité d'y resté indéfiniment puisque les ennemis vont soit venir de derrière soit ceux dont le joueur ce cache ce déplacerons pour pouvoir tirer. Certain point sont construit de manière stratégique pour donné un repos aux joueurs en offrant plus de couverture que d'autre mais ces points sont éloigné pour éviter un abus de ces derniers afin de ne potentiellement jamais mourir. Certain chemin sont plus à découvert mais permettent de rejoindre un autre point stratégique plus rapidement aux prix d'une possibilité de ce faire tirer dessus cependant des chemins protégé mais plus long relient les différents points stratégique laissant le choix du

risque aux différents joueurs.

Le niveau de détails de cette carte est semblable aux carte que l'on peux trouver dans la partie solo l'usage de caisses, de conteneurs ainsi que de différents autres objet rappelant la mode un joueur de notre jeux vidéo seront mis en place sur la totalité de la carte. Permettant de rappeler aux joueurs les différents passages des premiers niveaux du jeux. Mais l'accent est mis sur le positionnement des différents objets donnant l'impression que tout à été mis sens dessus dessous pour donner l'impression d'une grande attaque aillant créée un certain chaos au sein de la carte.

3.3 Ambiance sonore

3.3.1 Musique

Afin de proposer une expérience de jeu pertinente et en lien avec le contexte du jeu, nous avons sélectionné plusieurs musiques proposant une ambiance adéquate à des scènes de combats et d'infiltration volontairement tendues pour garantir une immersion totale du joueur dans le jeu.

Durant les scènes de combats, où le joueur doit être stimulé, la musique aura un thème électronique. En revanche, lors des phases d'infiltration, durant lesquelles le joueur sera concentré, la musique sera plus calme tout en restant en adéquation avec le thème général du niveau.

En revanche, Notre mode de jeu plus riche en action possèdera des morceaux ayant un tempo beaucoup plus soutenu et reposera sur des nuances de synthétiseurs et de guitare électrique.

Pour ajouter notre identité au jeu, nous allons composer deux thèmes originaux. Afin de compléter notre bande sonore, nous allons aussi utiliser des musiques libre de droits. Toutes musiques seront, tout comme les musiques précédentes, en lien avec l'ambiance sonore choisie par notre groupe.

L'implémentation du manager audio fut faites pour la deuxième soutenance mais subie quelques améliorations pour le rendu finale. Pour ce faire, la création d'une classe intermédiaire pour gérer les sources sur « Unity » était inévitable. La classe Son est cette classe qui ne comporte quasiment que des valeurs sérialisabe sur « Unity » (c'est-à-dire des valeurs que l'on peut changer dans « Unity « directement sans passer par un script). Elle comporte des variables comme le nom du son qui peut être une musique aussi (cette classe marche pour toute sorte de sons), un audio clip (c'est-à-dire un clip audio avec le chemin pour y accéder), une section volume réglable, une section pitch réglable, un booléen pour choisir si c'est une musique (c'est important de le savoir comme nous le verrons dans une partie ultérieure), un booléen « loop » (boucle) pour savoir si nous devons faire tourner la musique en boucle

ou non et enfin un paramètre que nous avons précédé de « HideInInspector » (Cache dans l'inspecteur littéralement) qui est la source audio qui va nous permettre d'utiliser toute les variables vu précédemment.

Ainsi, nous utilisons cette classe dans un autre script celui-ci appelé « AudioManager » (manager audio) qui va nous servir à implémenter tous les sons et musiques que nous trouverons dans notre jeu. Tout d'abord, on retrouve une liste de sons de la classe son vu ci-dessus qui va nous permettre de gérer chaque son, le mixer audio qui permet de gérer le volume du jeu dans les options, une variable « currentMusic » (musique courante) qui va nous permettre de savoir quelle musique est en train d'être joué sachant que celle-ci a une valeur nulle à son démarrage (aucune musique n'est lance avant que le jeu lui-même ne commence) et une variable instance pour que l'on garde le même audio manager de scènes en scènes. Ensuite on retrouve, dans la fonction « awake » (réveille) du script, une condition qui regarde s'il y a déjà une instance de manager audio, le cas échéant, l'audio manager courant la deviens et sinon on détruit cet objet (en l'occurrence le manager audio pour éviter des doublons). Ensuite on passe le manager audio en objet qui n'est pas détruit entre chaque scène pour éviter que la musique ne se coupe entre chaque niveau. Ensuite pour chaque son qui se trouve dans la liste des sons que nous avons, nous leur association une nouvelle source audio, puis respectivement, on associe le clip du son au clip de la source audio, le volume au volume de la source audio, le pitch au pitch de la source audio, la boucle a la boucle de la source audio et enfin on associe la source audio au mixer principal que nous avons vu en paramètre.

Ensuite dans ce script, on retrouve la fonction « Play() » (Jouer) qui prend une suite de caractère en paramètre et ne retourne rien. Elle cherche le son par rapport a son nom dans la liste des sons de la classe que nous avons vue ci-dessus et, si elle ne le trouve pas, affiche un message dans le livret de bord de « Unity » et sinon elle regarde si ce son est une musique. Si s'en est une, et qu'il y a déjà une musique en cours, alors on arrête la musique en cours et on la remet à zéro pour la prochaine éventuelle fois ou elle sera jouée puis on remplace la nouvelle musique à la place de la musique courante pour que celle-ci la devienne. Et enfin on lance la source qui est lie par la classe et la fonction réveille que nous avons déjà expliqué.

Pour la composition du thème principal nous avons, grâce au fait que je sois membre du Bureau De la Musique du campus, pu enregistrer avec une table de mixage, des amplificateurs et plusieurs guitares le thème principal de notre jeu. Nous nous sommes inspirés de thème simple que nous connaissons dans le thème de l'infiltration pour coller avec l'ambiance général du jeu et nous avons choisi de la mettre dans le menu principal des le lancement du jeu. Nous voulions faire court car quelque chose de court nous paraissait plus

facile à mémoriser sachant que le joueur n'allais pas rester indéfiniment dans le menu principal du jeu.

La sélection des musiques à mettre dans notre jeu c'est faite sur deux critères principaux :

— Les musiques devait être libres de droit,

— Les musiques devait convenir a l'ambiance générale du jeudi.

Avec ces deux critères, il ne restait déjà que peu de musiques qualités existantes sur internet et le choix était devenu beaucoup plus restreint. Malgré ces difficultés, nous avons réussi à trouver trois musiques qui correspondent aux critères que nous avons fixe. Voici ces musiques que l'on retrouve à certains endroits en fonction des niveaux que nous avons :

— Brutal Cycle (TeknoAXE)

— Waypoint J (TeknoAXE)

— Mission A (TeknoAXE)

Cet auteur nous permet d'utiliser ses œuvres gratuitement et librement. Il autorise une utilisation commerciale comme personnelle de ses œuvres.

3.3.2 Bruitages

En plus de l'ambiance sonore proposé par les musiques du jeu, nous avons décidé d'utiliser des sons spécifiques à chaques armes. Ces sons seront des enregistrements, pour le pistolet et le fusil d'assaut, de véritables armes. En effet, il nous est possible d'enregistrer des pratiquants d'armes à feu dans un stand de tir.

3.4 Menu du jeu

3.4.1 Menu principal

Le menu principal fût implémenté durant la première soutenance. Il est d'une complexité mineure mais me pris du temps car c'était la première fois que je tentais quelque chose sur « Unity ». Il est constitué d'un panel avec l'image de notre logo en fond, trois boutons et une barre en bas du panel avec le nom du jeu. Il fut ensuite modifié après plusieurs changements dans ce que nous voulions faire tout en restant dans le cahier charges évidement. Le menu comporte des scripts avec des fonctions qui sont associés aux boutons. Une « start » pour démarrer le jeu (qui a été modifié pour lancer le menu choix entre multijoueurs et solo), un script options qui lance le menu es options et un bouton quitter pour quitter le jeu.

3.4.2 Menu des options

Le menu des options a été fait en deux parties, une première pendant la première soutenance et une autre partie sur la dernière soutenance. D'un point de vue purement visuel, le menu est de couleur rouge foncé (qui rappelle le sang, l'infiltration est la dimension d'espionnage omniprésente dans notre jeu) avec une barre avec le nom du jeu avec la même police que le menu principal pour ne pas dépayser l'utilisateur, un bouton pour choisir si le joueur se met en plein écran ou non. Plusieurs résolutions en fonction de la capacité de l'écran de l'utilisateur, le choix des graphismes (très bas, moyen, haut et ultra) en fonction de l'appareil de l'utilisateur et enfin un curseur coulissant pour ajuster le volume (« slider »). Ces boutons et cases de choix sont de couleur blanches pour contraster avec le rouge et rappeler le noir et blanc que l'on trouve dans le fond du menu principal qui comporte une image en noir et en blanc. Enfin on retrouve deux images de pistoles en blanc toujours pour contraster et qui rappelle l'arme principale du jeu et celle qui est donnée de base. Enfin on retrouve l'option écrite dans la même police que celle du jeu en blanc entre les pistolets.

D'un point de vue du code, ce fut relativement difficile à implémenter car j'ai dû me renseigner sur plusieurs aspects différents du jeu. Pour les résolutions, je dois regarder quelle sont les résolutions de l'ordinateur du joueur et mettre celle qui sont à disposition dans la case pour choisir la résolution dans le start. Ensuite il y a une fonction qui sert à choisir la résolution en fonction de l'index qui correspond à la résolution choisie. Il y a un bouton fullscreen qui est lié à un toggle, qui change la fonction de plein écran grâce au script du menu, pour les graphismes, j'ai aussi mis plusieurs choix qui correspondent au graphisme du jeu. C'est encore une fois grâce à un script qui est dans le menu qui est lié aux paramètres de graphisme du jeu. Enfin le curseur coulissant est lié à l'Audio Mixer « Master » de notre jeu auquel chaque son est lié pour la sortie du son.

3.4.3 Menu de pause

Le menu de pause fut pour moi la plus grosse difficulté de ce jeu mais nous en parlerons dans une autre partie faite pour cela. Le menu pause a été implémenté avec succès de façon non triviale. Effectivement, ce menu se décompose en deux préfabriqués distincts : le panel qui affiche les boutons et le canvas qui fait en sorte d'être visible pour la caméra. D'un point de vue esthétique, j'ai choisi de mettre le fond du jeu plus sombre et d'affiché simplement les boutons en blancs, j'ai préféré faire quelque chose de sobre dans cette partie pour garder le côté réaliste du jeu et ne pas m'attarder sur

des détails esthétique car il y avait des problèmes techniques.

D'un point de vue du code, j'ai implémenté un script qui, si le joueur appuie sur la touche échap, lance la fonction pause et s'il réappuie sur cette même touche lance la fonction Resume du jeu. La fonction pause stop le temps dans le jeu, libère la souris pour pouvoir cliquer et enfin affiche les boutons tout en assombrissant le jeu en fond. La fonction résume, remet le temps, rebloque la souris et n'affiche plus le menu de pause. Le menu en lui-même comporte trois boutons. Le premier bouton resume lance la fonction resume expliquée ci-dessus, le second menu renvoie en menu tout en remettant le temps et enfin le bouton quitte quitte le jeu.

3.4.4 Menu lors de la mort

Le menu de mort a été implémenté pour le rapport finale. C'est un menu de la même couleur que le menu des options c'est-à-dire rouge fonce. Il comporte des écriteaux blancs. Un premier en anglais avec écrit « You died » (Vous êtes mort) et un second avec écrit « press f to restart » (pressez la touche f pour recommencer). J'ai dû faire un script qui y est associé qui, dès que l'utilisateur appuie sur la touche f, recommence le jeu a la première scène du mode histoire. J'ai choisi de faire un menu sobre pour encore une fois garder le cote réaliste et pour faciliter l'expérience de jeu. (il devient aisé de recommencer à jouer en appuyant simplement sur une touche)

3.5 Modélisation 3D

Notre jeu utilise massivement des objets 3D pour les cartes, les armes ou les ennemis. La majorité fut réalisé sur le logiciel de modélisation Blender. Ainsi, j'ai réalisé plusieurs modèles 3d «low poly» pour les décors et les armes. Ainsi, le fusil d'assaut, le pistolet et le pistolet mitrailleur sont modélisés. Mais aussi des éléments décoratifs tel que des caisses, des arbres ou des pierres. Afin de rendre les armes plus attractives, je me suis inspiré d'arme réelles : Le fusil d'assaut se base sur le M16A1, un fusil américain utilisé lors de la guerre froide. La seconde arme, le pistolet fut influencé par le SIG-SAUER P228. Enfin, le pistolet mitrailleur représente un UZI, reconnu pour sa fiabilité et sa petite taille.

3.6 Textures

Pour notre projet j'ai réalisé des images permettant d'être appliqué aux niveaux et aux objet de notre jeu. Ses images sont appelées textures.

La création de textures se fait en quatre étapes : La première étape est la prise d'image de référence. C'est à dire de prendre en photo des matériaux, des objets qui s'apparente à ce que l'on souhaite créer. Ces images de références peuvent être prises dans différents lieux, comme une forêt ou dans un bâtiment abandonné.

La deuxième étape est le traitement des images de références. À l'aide d'une suite logicielle adapté, comme PhotoShop ou GIMP, il est nécessaire de gommer certaines imperfections de ces images tel que de la saleté ou des insectes.

Viens ensuite une modification importante de ces images : modification en textures dite « Seamless». Une texture dite « Seamless » est une image qui peut être placée côte à côte avec elle-même sans créer une limite perceptible entre deux copies de l'image. Pour réaliser ceci, plusieurs choix s'offre à nous. Il est possible de réaliser ce traitement par un ou plusieurs algorithmes ou de retoucher l'image manuellement afin de créer l'effet souhaité. La création à l'aide d'algorithmes peut être plus aisé et plus rapide mais a comme principal inconvénient une qualité dépendant beaucoup de la prise d'image initiale. Une création manuelle quant à lui est bien évidement bien plus lente mais est souvent de meilleure facture. Pour créer nos propres textures, j'ai combiné ces deux méthodes. Une première tentative est faite manuellement et un algorithme se charge d'adoucir les bordure.

Enfin, la retouche finale ou la création de texture plus spécifique. Une fois notre texture prête à l'emploi, Nous pouvons ajouter quelques effets supplémentaires comme une modification de la couleur. Pour terminer, il est possible, comme évoqué précédemment, de créer des textures plus spécifiques qui vont permettre de donner des informations supplémentaires à notre texture. Ainsi, nous pouvons concevoir une texture de relief (Appelé « Bump map ») qui permet de donner apparence rugueuse à une matière à partir d'un modèle en 3D de notre texture par exemple. (Voir figure) Il en existe d'autre tel qu'une texture spéculaire (« Specular map ») qui permet d'indiquer la réflexion de la lumière. Toutes ces informations donneront un aspect plus réaliste à nos textures une fois incorporées en jeu. Plusieurs textures sont mises à disposition pour Jean-Philippe et Alexandre : Trois textures représentant du béton, deux de surfaces métallisés, une de bois et trois de murs. Toutes disposant d'une texture de relief à l'exception des surfaces de métal, possédant chacune une texture de relief et d'une texture spéculaire.

3.7 Gameplay

Le joueur peut se déplacer librement sur une carte à l'aide de touches directionnel et les collision joueur-carte-objet. Le joueur peut aussi utiliser

sa souris afin d'explorer son environnement en trois dimension. De plus, le joueur a la possibilité de neutraliser des ennemis à l'aide d'armes à feu ramassés dans les niveaux. Une arme ramassée est ajouté dans l'inventaire invisible du joueur. S'il possède déjà l'arme qu'il ramasse, les munitions seront ajoutés dans une des trois catégories d'arme disponibles (Légère, Moyenne ou Lourde). Chaque arme possède des dégâts, une cadence de tir et un nombre de munition qui leur sont propre. Afin de déterminer si le joueur touche un ennemi lors du tir, les armes utilisent le principe de 'Raycast' ou le lancer de rayon en Français. Dès que le joueur tire, on décrémente la quantité de munition disponible pour cette arme puis on vérifie si le 'Rayon' envoyé ne touche pas un mur ou tout autre objet qui n'est pas un ennemi. Cette vérification est faite en comparant l'étiquette de l'objet en question avec l'étiquette 'Enemy'. Si l'objet touché est un ennemi, nous appelons dans cet enemy une fonction permettant de décrémenter la vie de cet objet en fonction des dégâts de l'arme utilisée.

Type de Soldat	**Arme**	**Vie**	**Vitesse (par rapport au joueur)**	**Apparence**
Léger	Pistolet	75	125%	Costume
Moyen	Pistolet-mitrailleur	100	100%	Kevlar
Lourd	Fusil d'assaut	150	50%	Tenue blindée

TABLE 1 – Caratéristiques des gardes

3.8 Armes

Les armes utilisables dans le jeu seront toutes dotées d'un silencieux excepté le fusil d'assaut. Le silencieux ne s'utilisera pas comme dans la réalité, il permettra d'éliminer les gardes sans se faire repérer.

Nous avons choisi les armes suivantes et leurs caractéristiques propres :

Arme	**Dégâts (Points de vie)**	**Précision**	**Taille du chargeur**
Pistolet	30	100%	10
Pistolet-mitrailleur	40	70%	30
Fusil d'assaut	50	50%	20

TABLE 2 – Caratéristiques des armes

3.9 Multijoueur

Le multijoueur a pu être implémenter dans le jeu après avoir résolu de nombreux problème qui sont survenue les uns derrière les autres. Tout d'abord le multijoueur à du être mis de côté le temps de l'implémentation des principales fonctionnalités du jeu tel que le déplacement et le moyen de courir, la fonctionnalité permettant de tirer ainsi que la création de l'intelligence artificielle. Après que ces différentes fonctionnalités aient été mise en place je me suis mis à la création du multijoueur mais pour cela j'ai du utilisé un asset permettant de gérer le mode multijoueur du jeu. Cet asset est en fait développé par photon et se nomme «PUN 2» ce dernier ajoute ainsi plusieurs commandes permettant de gérer le multijoueur. Il me fallait d'abord comprendre comment les différents joueur allait se connecter ensemble. Ainsi, j'ai compris qu'il me fallait d'abord connecter le client au serveur, appelé «Master» ou serveur principal, pour ce faire j'ai utilisé un champ d'entée pour que le joueur puisse mettre son pseudonyme avant d'entrée dans le serveur «Master», une fois que le joueur à entré son pseudonyme ce dernier peux cliquer sur un bouton pour se connecter au serveur principal. Une fois entrer dans le serveur «Master», ce serveur lui permet alors d'accéder au différentes salles de jeux par le biais d'un menu répertoriant toute les salles de jeux, Il permet aussi de rejoindre une salle de jeux aléatoire grâce à un bouton et si aucune salle n'existe ce bouton créer automatiquement une salle de jeux ,un champ d'entrée permet de donner un nom à la salle que l'on souhaite créer une fois le nom écrit dans le champ d'entré un bouton se situe juste en dessous permettant de lancer la création de cette salle de jeux et de la rejoindre dans le même instant. Ce menu à causé énormément de souci puisque ce dernier n'ajoutait pas les salle de jeux crée à la liste des salles de jeux . Ce souci ayant été réglé par le biais de différentes fonction permettant de détecter une nouvelle salle de jeux et d'ainsi la rajouter à la liste permettant une connexion simple avec d'autres joueurs. Les salles de jeux quant à elles, elles sont limité à un nombre maximum de joueur qui n'est autre que quatre. Une fois une salle de jeux rejoins on peux voir les pseudonyme des différents joueurs afin de savoir avec qui l'on joue. Seul l'hôte de la salle de jeux peux voir et utiliser un bouton permettant de lancé la partie, les autres joueurs voient un message leur indiquant de patienter que l'hôte de la salle de jeux lance la partie. Une fois en jeu les joueurs se retrouvent dans une carte ressemblant à une arène en forme d'hexagone basé sur le thème d'un hangar ou se trouve des armes et des ennemis. Ainsi, les joueurs devront survivre en tuant le plus d'ennemis possible afin d'augmenter le score de l'équipe chaque joueur pourra ramasser ces armes afin d'accomplir l'objectif. Si tout les joueurs viennent à mourir la partie ce termine et affiche le score de

l'équipe ,proposant ensuite au joueur de quitter ou à l'hôte de recommencer la partie. Les ennemis arriveront alors à chaque vague mais aucun moyen de récupérer de la santé ne sera mis à disposition des joueurs afin de rendre le jeu plus intense ,forçant ainsi le joueur à essayer de ne pas ce prendre de dégât ou bien d'en prendre le moins possible, mes sans pour autant laisser les joueurs déjà mort ces derniers réapparaitrons lors de la manche suivante. Le joueur sera alors plus méfiant et le jeux sera dotant plus prenant puisque le joueur devra régulièrement chercher des munitions sur les ennemis pour pouvoir continuer à se défendre.

3.10 Site internet

Le site internet de notre projet est conçu en partie par un outil de génération de site web statique complétement écrit en Python. Il nous suffit de créer une page de base et le contenu, écrit en Markdown, est converti en HTML et inséré dans cette page. Le Markdown est un langage de balisage simple conçu par John Gruber en 2004. Il permet d'ajouter facilement du formatage, des liens ou des images à du texte brut. Il fut conçu pour être une alternative plus facile d'accès que l' HTML et d'écrire du contenu sans avoir aucune connaissance en HTML. Ce langage est populaire et utilisé sur les blogs ou encore les forums. Ainsi pour notre site il n'est pas nécessaire de modifier le code source d'une page html afin d'écrire du contenu. Il suffit juste de modifier le fichier Markdown afin d'obtenir une page différente. Comme dit précédemment, un texte formaté en Markdown nécessite un interpréteur générant du code HTML à partir de celui-ci. J'ai choisi un interpréteur connu pour sa simplicité et sa vitesse : Mistune. Mistune est un interpréteur Markdown écrit en Python et contient quelques améliorations qui enrichissent Markdown. Il est aussi possible de modifier son comportement et d'ajouter nos propres fonctionnalités. Afin d'embellir notre site internet, nous avons utilisé un thème conçu initialement avec Hugo, un autre générateur web statique. Amaury, après mûres réflexions, a modifié le thème de notre site internet pour le faire fonctionner avec notre propre générateur.

Notre site internet est hébergé sur Dyjix, un hébérgeur à but non lucratif. Nous l'avons choisi car il dispose des offres étudiantes intéressantes. Pour le nom de domaine, nous avons choisi PulseHeberg. Celui–ci propose un nom de domaine « .fr » abordable. Pour mettre en ligne notre site généré avec notre outil (voir la partie précédente), nous avons utilisé le protocole FTP et plus précisément un client FTP appelé filezila. Après avoir généré notre site, nous l'avons mis sur Dyjix et ainsi il était en place et accessible directement. Nous avons adapté le thème téléchargé pour qu'il corresponde à notre vision du jeu et nos attentes, ainsi chaque annonce correspondante à nos soutenances

peut être publiée et nous avons une partie liée contact.

3.11 Logiciels Utilisés

— Unity : Le moteur de jeu utilisé,
— Blender : Logiciel de modélisation 3D,
— Audacity : Logiciel d'édition sonore,
— Jetbrains Rider® : IDE permetant d'écrire du code C# pour Unity,
— GIMP / Krita : Logiciels d'édition d'images,
— Firefox / Google Chrome : Navigateurs internet,
— Linux Multi Media Studio / Ableton Live® : Logiciel de composition musicale.

4 Avancement du projet

4.1 Tableau de la répartition des tâches

Catégories	Alexandre	Amaury	Benoît	Jean-Philippe
Intelligence artificielle				
Pathfinding	**X**	+		
Détection du joueur à distance	**X**	+		
Niveaux				
Architecture	**X**	+	+	**X**
Détails	**X**	+	+	**X**
Gameplay				
Déplacement du joueur	+		**X**	
Gestion des armes	+		**X**	
Modélisation 3d				
Armes			**X**	+
Ennemis			**X**	+
Aspect sonore				
Effets sonores		+	**X**	
Musique		**X**	+	
Autre				
Multijoueur		+		**X**
Menu		**X**	+	

TABLE 3 – Répartition des tâches

Légende :
— X : Responsable(s)
— + : Suppléant(s)

4.2 Avancement estimé des tâches par période de soutenances

Catégories	Première soutenance	Deuxième soutenance	Soutenance finale
Intelligence artificielle			
Pathfinding	20%	50%	100%
Détection du joueur à distance	60%	90%	100%
Niveaux			
Architecture	30%	60%	100%
Détails	10 %	65%	100%
Gameplay			
Déplacement du joueur	70%	90%	100%
Gestion des armes	30%	70%	100%
Modélisation 3d			
Armes	20%	60%	100%
Ennemis	20%	60%	100%
Aspect sonore			
Effets sonores	30%	60%	100%
Musique	20%	70%	100%
Autre			
Multijoueur	10%	40%	100%
Menu	5%	60%	100%
Site internet	10%	60%	100%

TABLE 5 – Répartition des tâches entre période de soutenance

Intelligence artificielle :

Pour la première soutenance, nous comptons programmer l'intelligence artificille à 40%, ce qui correspond à un ennemi immobile, qui tire sur le joueur s'il se situe directement devant lui. Le palier des 80% correspond à l'ajout d'un cône de détection des ennemis (cf intelligence artificielle), les gardes pourront se déplacer mais pas autour du joueur durant les combats, ils pourront également viser le joueur. Enfin, lors de la soutenance finale, les ennemis pourront se déplacer le long d'un axe lors des combats, tout en tirant sur le joueur.

Niveaux :

Une avancée de 30% dans l'architecture globale correspond aux deux premiers niveaux du jeu, sans détails ni éléments décoratifs, composés de blocs. Une progression de 60% dans les niveaux correspond à l'ajout de trois autres niveaux. Aussi, chaque niveau sera partiellement détaillé, les trois premiers le seront davantge. Nous ajouterons tous les éléments décoratifs manquant, ainsi que les détails dans les niveaux, il ne sera plus question de blocs.

Gameplay :

Nous prévoyons d'avancer de 70% pour la première soutenance dans les déplacements du joueur, ce qui signifie que le joueur pourra anvancer en marchant (vitesse moyenne) ou en courant (vitesse plus élevée). Nous prévoyons d'ajouter des animations de course et de marche pour la deuxième soutenance, ce qui constitue les 20% supplémentaires. Enfin, nous ajouterons la possibilité de s'accroupir, ainsi que l'animation correspondante.
L'implémentation du gameplay contient aussi la gestion des armes. En effet, nous comptons rendre possible l'utilisation du pistolet par le joueur,ce qui est équivalent aux 30% indiqués dans le tableau pour la première soutenance. Les 70% prévus pour cette section correspondent à l'ajout de toutes les armes, ainsi qu'à leur implémentation (il sera possible de tirer avec). Cependant, il ne sera pas possible de recharger une arme. C'est ce que représentent les 100% annoncés pour la soutenance finale.

Modélisation 3D :

Nous avons pour objectif de modéliser une arme basique (un pistolet), composé de cubes. Cela représente 20% de la modélisation des armes pour la première soutenance. Les 60% indiqués dans le tableau pour la deuxième soutenance correspondent à des armes plus détaillées, mais sans animations. Ces animations seront ajoutées pour la soutenance finale, et termineront cette sous-section.
Les ennemis, dont la modélisation générale sera achevée à 20% pour la première soutenance, seront peu détaillés, des «stickmen en 3D». Lors de la deuxième soutenance, les ennemis seront plus détaillés, leur modélisation sera presque terminée. Enfin, l'ajout d'animation pour la soutenance finale correspond aux 100% indiqués dans le tableau.

Aspect sonore :

Nous avons décidé d'implémenter, pour la première soutenance, un son commun à toutes les armes présentes. Cette avancée représente les 30% des effets sonores. Lors de la deuxième soutenance, nous aurons distingué les bruits de pas de course et de marche. Aussi, toutes les armes auront leur prorpe bruitage. Il ne restera que le bruitage du rechargement, qui sera ajouté pour

la soutance finale (et représente donc les 100%).
Nous ajouterons à notre jeu deux musiques pour la première soutenance : une musique d'infiltration et une musique de combat. Cela constitue 20% de la musique du jeu. Nous aurons ajouté deux autres musiques pour la deuxème soutenance. Enfin, Lors de la soutenance finale, toutes les musiques seront prêtes, en accord avec le gameplay du joueur (la musique changera si le joueur se fait reprérer), ce qui achève cette partie.

Autres :
Lors de la première soutenance, nous aurons fait 10% de la section multijoueur, ce qui correspond à se déplacer sur une carte avec un autre joueur, localement. Nous aurons ajouté la possibilité de tirer sur des ennemis fixes pour la deuxième soutenance. Enfin, le mode multijoueur sera terminé lors de la soutenance finale et possèdera les caractéristiques décrites dans la section multijoueur. En parallèle, nous implémenterons le menu à 5% pour la première soutenance : deux boutons simples, à savoir «Play» et «Play multiplayer», qui indiqueront respectivement le mode solo et le mode multijoueur. La possibilité de sélectionner un monde dans le menu sera ajoutée pour la deuxième soutenance, ce qui correspond aux 60% indiqués dans le tableau. Enfin, nous aurons ajouté le menu options, qui permettera de faire varier différents paramètres tels que la qualité graphique et la sensibilité de la souris.

4.3 Avancement réel lors des périodes de soutenances

Catégories	Première soutenance	Deuxième soutenance	Soutenance finale
Intelligence artificielle			
Pathfinding	30%	50%	100%
Détection du joueur à distance	60%	95%	100%
Niveaux			
Architecture	30%	60%	100%
Détails	10 %	65%	100%
Gameplay			
Déplacement du joueur	70%	90%	100%
Gestion des armes	30%	70%	100%
Modélisation 3d			
Armes	15%	60%	100%
Ennemis	15%	60%	100%
Aspect sonore			
Effets sonores	30%	60%	100%
Musique	20%	70%	100%
Autre			
Multijoueur	10%	40%	100%
Menu	30%	60%	100%
Site internet	10%	60%	100%

TABLE 7 – Répartition des tâches entre période de soutenance

5 Difficultées techniques

5.1 Unity

Cette section aborde les problèmes rencontrés sur le moteur de jeu que nous utilisons pour notre projet.

5.1.1 Gestion des matériaux

Unity possède un système de gestion des matériaux, permettant de définir à une surface virtuelle comment celle - ci doit être calculée. Il est possible d'ajouter des informations comme la rugosité, les couleurs ou la texture appliqué ce matériau. Nous utilisons ces matériaux pour les niveaux de notre jeu. Lorsque nous plaçons un matériau sur un modèle ou une portion de niveau, celui - ci se retrouve appliqué sur l'objet sélectionné. Malheureusement, si nous décidons de changer les propriétés de ce matériau sur un objet précis, le matériau est automatiquement modifié sur tous les objets utilisant ce matériau. Ainsi, nous sommes obligé de créer plusieurs dizaines de matériaux ayant un seul paramètre modifié. Créant dans le processus plusieurs duplicatas inutiles.

5.1.2 Unity Collab et Git

Le code source de notre jeu était centralisé sur la plateforme collaborative Gitlab disposant d'une intégration Git. Après plusieurs mois à travailler sur notre projet, nous découvrons avec stupéfaction que des fichiers disparaissent ou se retrouvent irrécupérables. Lors de nos recherche et plusieurs messages d'erreurs, nous apprenons que l'extension de Git que nous utilisons, à savoir Git-LFS, posait problème. Nous avons ainsi dû réinitialiser notre dépôt plusieurs fois et installer Git-LFS sur les postes dont nous utilisons Unity. Après de s'être concerté avec toute l'équipe, nous avons décidé de migrer sur une autre plateforme, proposant les mêmes services que Gitlab mais ayant une bande passante plus élevée et disposant une meilleure intégration et gestion de gros fichiers. Ne disposant que de peu de temps, nous nous sommes résigné de changer pour "Unity Teams". "Unity Teams" est un système regroupant des services mis en place par les développeurs du moteur permettant de concevoir notre jeu, il dispose d'un système de collaboration étroitement intégré au moteur. Limité à trois personnes dans sa version la plus basique et disposant d'un gigaoctet de stockage pour tout le projet, nous avons donc acheté la version plus avancée de ce système. Cette version plus avancée dispose de 25 gigaoctets de stockage et la possibilité de réaliser des compilations en ligne.

Une plaisante surprise fut que ce service ne donne pas accès à une place supplémentaire pour l'édition du projet. Nous avons attaché en supplément le droit à un autre développeur de travailler sur celui - ci. Nous pouvons voir chaque modification apportée au projet presque en temps réel. Cet outil n'est pas exempt de défauts. Pour commencer, il est courant que l'outil de collaboration cesse de fonctionner, affirmant que le projet n'utilise pas le système de synchronisation que vous avons préalablement activé. Nous devions re - télécharger intégralement le projet, réactiver la synchronisation et ajouter manuellement nos modifications précédentes. D'un autre côté, la collaboration ajoute un temps de chargement supplémentaire non négligeable lors de l'ouverture du projet. Ceci peut allonger le chargement du projet de une à deux minutes pour un petit projet à plus d'une dizaine de minutes pour un projet plus conséquent.

5.1.3 Support de Linux par Unity

J'ai rencontré un nombre incalculable de problèmes liés au fait d'utiliser « Unity »sur Linux. Je tiens à préciser que j'avais la même version que mes collègues de travail c'est dire Unity 2018.3.0. Je tiens aussi a dire que je suis resté sur Linux car mon Windows sur mon ordinateur portable ne me permettais pas d'avoir une version assez récente de Windows pour avoir « Unity », ainsi j'étais contraint soit de travailler chez moi ce qui ne colle pas bien avec la définition de travail de groupe et qui causerait plus de problème que la solution que j'avais essayée d'adopter au début du projet c'est-à-dire utiliser « Unity » sur Linux.

Tout d'abord l'incompatibilité avec la version Windows était déplorable, au départ quand nous utilisions git, a chaque fois que je faisais un pousser, il y avait systématiquement un problème même si je n'avais rien changé. Les changements que mes collègues effectuait ne marchaient pas la plupart du temps, malgré les conseils de Marc en début d'année « Unity » ne fonctionne pas bien sur Linux.

Ainsi après plusieurs de dizaines de sessions d'énervements ou je passais plus de temps à essayer de voir pourquoi « Unity »ne marchait pas plus qu'à coder, et à m'énerver moi et mes collègues que j'ai choisis d'utiliser « Unity » sur Windows chez moi sur mon ordinateur fixe. Ainsi tous les problèmes lié a ce que j'ai évoqué avant était réglé et j'ai enfin pu me mettre à travailler de façon efficace mais avec le désavantage d'être chez moi et donc de devoir contacter mes membres via discord ou par téléphone pour communiquer avec eux sur les sujets qui nécessitaient de parler pour une meilleure compréhension.

5.1.4 Les «prefabs»

Au début de mon travail sur « Unity », je n'avais pas particulièrement besoin d'utiliser de prefab sur le logiciel, les "Prefabs" sont un regroupement d'objets Unity. Les menus ne sont à faire qu'une seule fois dans la scène du menu. Quant à l'Audio manager j'ai été contraint d'utiliser des prefab pour le rendre accessible dans chaque scène. Ainsi cette prefab du manageur audio ne m'a posé aucun problème sûrement dus au fait que c'était un objet qui dépendait uniquement d'un script que j'avais écrit et pas quelque chose qui existait déjà saur « Unity ». Puis je fis le menu de Pause qui doit être implémenté dans chaque scène. C'est là que mon plus gros problème lié a « Unity »tout court apparut car il a persisté même quand j'étais passé sur Windows pour travailler sur le projet.

Pour faire ma prefab du menu de pause je suis allié sur une map que nous avions fait et je fis comme un menu classique, après m'être renseigne sur quelle son les valeurs lie a la pause, et ainsi ce menu marcha sur cette map. Avec une joie non négligeable je m'empressai de le mettre dans les prefab et de faire comme « d'habitude » mais dès que je le mis sur une deuxieme map... Se fis le drame : les boutons ne fonctionnaient plus. Je me dis donc que ;'erreur venait de moi et que ce n'était pas possible, je me pris donc du temps à essayer de comprendre pourquoi une prefab de boutons ne fonctionnait pas et je lus toute la documentation « Uniy » a ma disposition. J'essayai pendant plusieurs jours et même semaines avec des choses différentes qui ne fonctionnait pas. Je pris donc l'initiative de demander a mes camarades de m'aider, il n'avait jamais eu ce problème et j'ai même recommencé devant eux le menu pour leur montrer le problème. Je demandai à quasiment tous les groupes de l'aide me disant que ce ci aurait eut le même problème de moi car j'étais assez désespéré. Aucun n'avait eut ce problème et n'était capable de m'aider...

Enfin c'est en créant un prefab de chaque parties indépendantes que je vis que se jetait pas les boutons qui posaient problèmes mais le canvas qui servait à faire le menu qui quand on mettait en prefab, désactivait les boutons. Ainsi en essayant plusieurs de façon aléatoire, je pris un fichier sur la scène (dont je n'ai jamais entendu parler sur la doc ou sûrement celle-ci n'était pas la bonne version car celle de notre version n'existe pas...) appelé SystemEvent et je le mis dans le prefab et comme par magie, le prefab marcha. C'est comme cela que je fus confronté au plus gros problème que j'ai eu dans mon projet dont l'erreur ne venait pas de moi mais de « Unity » lui-même qui a sûrement besoin d'améliorations et je n'hésiterais a en faire part aux développeurs dès la fin du projet.

5.2 Le multijoueur

Problèmes engendrés durant le projet pour le multijoueur ainsi que les explication qui vont avec ces problèmes :

— Impossibilité d'implémentation au départ du projet, il était impossible d'implémenter le multijoueur au début du projet puisque les fonctions qu'il devait utilisées n'était pas entièrement terminé puisque le mouvement du joueur à été revue plusieurs fois ainsi que le déplacement de l'intelligence artificielle qui a été en permanence améliorer mais qui n'était pas utilisable pour le multijoueur au tout début.
Une première méthode pour régler le problème aurais été que les autres membres du groupe expliquent leurs codes et essayent de le modifié de manière à ce que le multijoueur ne soit pas impacté par les changements et que l'au contraire ces derniers soit plus pratique à utiliser , mais ceci pouvant ralentir leur progression dans leurs travaux nous avons opté pour une autre méthode plus approprié. Pour ce faire je suis allé voir chacun des codes des membres du groupe qui était en rapport avec la méthode de jeu qui est aussi utilisé en multijoueur afin de l'implémenter de sorte à ce que les changement soient directement pris en compte par le multijoueur.

— Aucun personnage créé lors du lancement de la partie, les personnages n'était pas généré lors du lancement de la partie car la référence des objets créés en multijoueur se trouve dans un dossier différent que ceux créés en solo.
L'utilisation d'une fonction de photon «PUN 2» permettait d'initialiser des objets tels que le personnage du joueur mais pour ce faire il était nécessaire recréer un objet, étant un regroupement d'objet prédéterminé avec leurs fonctions déjà misent en place, du personnage dans un autre dossier, le dossier «Référence», permettant ainsi à photon de récupérer l'objet souhaité pour le créer à l'endroit voulu. J'en ai profité pour créer un système de génération aléatoire autour d'un point mais restant sur le même axe de hauteur permettant ainsi à photon de générer les différents personnages des joueurs sans qu'il se rentre dedans.

— Problèmes de synchronisation lors de la création des personnages, l'usage d'une instanciation classique des personnages ne permettait pas de les faire apparaitre de manière synchronisé cela force l'usage de fonctions particulière vis-à-vis du multijoueur proposé par l'asset de photon qui est «PUN 2».
L'utilisation de la fonction précédemment citée permettant de générer les différents personnages ainsi que la modification d'autres fonctions

permet aussi d'attribuer un personnage au client de chaque joueurs permettant ainsi à chacun de contrôler uniquement son personnage les personnages étant créées sont en même temps synchronisé entre tous les clients permettant de voir exactement ce que les autres joueurs font sans avoir de décalage.

— Problèmes de synchronisation lors du déplacement des ennemis, les ennemis ne se déplacent pas de la même manière chez chacun des joueurs ceci étant dû au manque de synchronisation pour régler ce problème il faut alors modifier la fonction originale pour permettre à photon de pouvoir synchroniser lui-même les différents objets tel que le déplacement.
L'utilisation d'autre fonctions de photon permettent de synchroniser différentes actions telle que le mouvement des ennemis ces derniers font alors le même mouvement chez tous les clients de tout les joueurs de manière à ce que personne ne voient un ennemi à un endroit ou d'autre ne le voient pas, cependant les fonctions de photon ne font pas tous il faut alors modifié légèrement le code des ennemis afin que les déplacements se font vers le même endroit chez chacun des clients.

— Problèmes de synchronisation pour les dégâts des ennemies fait aux ennemis, de la même manière les dégâts aux ennemis ne sont pas retranscris chez tous les joueurs de la partie ne permettant pas la destruction de l'entité lorsque celle-ci n'a plus de point de vie chez un joueur. L'entité disparaitra chez le joueur qui la tué mais pas chez les autres joueurs.
L'utilisation des même types de fonction et nécessaire pour synchroniser les dégâts fais aux ennemis pour ce faire ce n'est pas les fonctions lié au ennemis qui sont modifié mais celle que le joueur utilise pour pouvoir tirer permettant ainsi de synchroniser le fait que le joueur à retirer des points de vies à l'ennemie en question, l'ajout d'un compteur d'ennemis tué et aussi ajouté à ce niveau afin de pouvoir afficher un score à la fin de la partie en multijoueur.

— Problèmes de synchronisation des armes possédées, lorsqu'un joueur entre en la possession d'une arme les autres joueurs ne voient pas l'arme que le joueur précédent a pris en main ceci est du à une liaison entre la caméra qui a lié à elle l'affichage des armes, celle-ci étant désactivé pour les personnages que le joueur ne contrôle pas.
La modification du préfab du personnage pour le multijoueur permet alors l'affichage de l'arme que le joueur possède permettant ainsi aux autre joueur de voir l'arme que ce dernier à en main et aussi de voir lorsque ce dernier change d'arme afin de ne pas être perdu afin de comprendre ce que font les autres joueurs et rendre le jeu un peu plus

immersif.

— Aucun personnage visible, les personnages n'avaient aucun corps visible rendant impossible les tests nécessitant la visibilité de minimum deux personnages.
La visibilité des personnages à été modifié par le biais de l'ajout d'un cor visible qui est désactivé pour le joueur possédant le personnage afin de ne pas obstruer sa caméra dans le même temps chaque joueur aura un signe distinctif permettant de se reconnaitre entre eux et de moins se confondre.

— Perte de l'avancement lors du changement de certaine méthode dans le déplacement, le personnage multijoueur étant géré différemment que le personnage en jeu solo, si ce dernier était modifié une modification différente était nécessaire pour les personnages multijoueurs que ce soit dans le code ou dans la préfab du joueur multijoueur.
Pour reprendre l'avancement fait il fallait alors refaire entièrement le préfab du personnage que photon prenais en référence car ce dernier était devenu obsolète et devait être entièrement revue si l'on voulais le faire fonctionner de manière correcte cependant cela prenais beaucoup trop de temps alors on a recréé le préfab du personnage multijoueur en reprenant le préfab utilisé pour le solo permettant ainsi d'avoir moins de changement à faire pour l'adapté au multijoueur.

— Perte due à une modification de fichier créer par « Unity Collab », un fichier ayant été modifié de manière non intentionnel par un membre du groupe suite à un envoi de ces information par le biais de « Unity Collab ».
Pour remédier à la perte d'avancement du à Unity Collab nous avons du rechercher dans chacune des précédentes versions que nous avons envoyé à Unity Collab afin de trouver celle qui nous faisait défaut avec un fichier important modifié de manière involontaire. Une fois de retour sur cette version nous avons dû refaire entièrement tout les progrès que nous avions implémentés depuis la modification de ce fichier nous forçant à tout revoir à chaque fonction pour savoir si l'on l'avait modifiée ou non. Une fois de retour à notre avancement initial nous prenons la précaution de vérifier les fichiers modifié à chaque fois que l'on veux envoyé notre progression à Unity Collab.

5.3 Autres difficultés rencontrées

Alexandre MOURERE

En tant que responsable de l'intelligence artificielle, il était de mon devoir d'implémenter la possibilité de faire des rondes pour les ennemis à travers la carte. Cette mission ne s'est pas effectuée sans difficulté dans la mesure où les techniques que j'ai utilisées entrent en conflit avec celles utilisées par Benoît MALHOMME dans l'implémentation du tir des ennemis. En effet, j'ai utilisé un système de tags pour détecter le bon ennemi afin de permettre aux ennemis d'effectuer des rondes indépendantes et propres à chaque ennemi. J'ai donc dû modifier le tag des ennemis pour l'associer à un chemin de ronde précis. Or, Benoît MALHOMME utilisait le tag de tous les ennemis pour pouvoir tirer dessus. Le tag étant modifié, nous ne pouvions plus tuer les ennemis. Afin de pallier le problème, j'ai modifié son script pour faire en sorte que le tire ne blesse l'ennemi en question que si son tag est différent de « null » ou « untagged », ce qui correspond aux éléments neutres du jeux, tels que les murs et les détails par exemple. En effet, si le tag d'un objet est différent de « untagged » et de « null », il s'agit soit d'un joueur, possédant le tag « Player », soit d'un ennemi. Puisque nous jouons en mode solo, il n'y a qu'un «Player». Ainsi, il s'agit nécessairement d'un ennemi. Ceci a été mon raisonnement me permettant d'implémenter correctement les rondes et le fait de pouvoir tuer les ennemis dans le jeu.

5.3.1 Menu principal

Le plus difficile dans la création du menu fut le travail sur notre logo pour le mettre à la bonne taille ainsi que le mettre en noir et blanc tout en correspondant avec notre vision esthétique et pratique de notre jeu vidéo. J'ai utilisé « GNU Image Maniplulator Program » pour faire ces travaux, un logiciel dont j'ai dû apprendre les outils lors de ces travaux.

5.3.2 Menu des options

Les problèmes que j'ai rencontrés étaient liés à la documentation d'Unity qui est difficile à trouver pour la version que nous utilisons car Unity change constamment. Malgré ce problème, je fus en capacité de créer ce menu d'options qui à une utilité non négligeable dans ce jeu et ramène un peu de customisation.

5.3.3 Menu de pause

Sur Linux, le texte ne s'affiche pas correctement. Le problème est sûrement lie à Unity et il n'y a pas grand-chosè à faire si ce n'est de le refaire sur Linux... Mis à part ce problème, rien ne fut particulièrement dur mis à part un problème majeur dont nous parlerons dans la section technique de notre rapport.

5.3.4 Menu lors de la mort

Au moment où l'implémentation de ce menu a été effectué, nous étions sur « Unity colab ». État concentré sur mon travail, je fis un premier changement en utilisant cet outil, or le menu n'était pas totalement implémenté car il manquait un paramètre à rentrer dans « Uniy ». Mes collèges ont donc téléchargé la version que j'avais effectuée mais qui mettais un bogue dans le jeu et qui les empêchaient conséquemment de travailler. Après quelques remarques de mes camarades, j'ai fixé le problème et tout est rentré dans l'ordre.

5.3.5 Implémentation du système audio

J'ai rencontré plusieurs problèmes lié au manager audio car évidemment la première fois que j'ai voulu le tester, aucune musique n'est sortie. Après plusieurs essaie infructueux, j'ai eu l'idée de regarder comment fonctionne l'audio en général sur «Unity» ce qui me permis de savoir que l'on avait besoin d'un audio mixer pour que le volume sorte (ce qui n'était pas sur la documentation que j'avais consultée ultérieurement).

Ensuite quand il y a eut du son, il était de volume à peine audible. Cela posait problème et après avoir monté tous les volumes possible et inimaginable sur « Unity », je me résigna à laisser tomber. C'est enfin a tête reposée que plus tard dans l'avancement du projet que je vis que tous les pitchs de mes testes audio était nulles. Je les changea et je venais de passer d'un audio manager avec un son à peine audible a un manageur audio fonctionnelle est efficace.

5.3.6 Implémentation du thème principal du jeu

L 'implémentation du thème principal du jeu fut trivial de par l'implémentation de notre manageur audio. Il a suffi de mettre la commande jouer du manageur audio avec le nom du thème principal que nous avons mis dans la liste des sons pour le lancer et peut être finalement lancé.

6 Fonctionnalités annexes

6.1 Compilation en ligne

Afin d'accélérer le développement de notre jeu en réduisant le nombre de test nécessaires au bon fonctionnement de celui - ci nous utilisons des services d'automatisation de compilation en ligne appelé «Unity Cloud Build». Le jeu est donc compilé et distribué automatiquement, réduisant ainsi le travail manuel et les interventions pour son déploiement. Son principal avantage est de vérifier en continu l'intégrité de notre projet, c'est à dire de déceler de potentielles erreurs introduit par les modifications fréquentes de notre jeu.

Son fonctionnement repose sur la détection des changement apporté au projet sur un logiciel de gestion de versions tel que Git, Mercurial, SVN (Appelé aussi «Subversion») ou encore «Unity Collab». Dès qu'un changement est réalisé, le dossier de notre projet est envoyé aux serveurs de compilation qui se chargent d'assembler l'exécutable de notre jeu de la plateforme de notre choix supporté par le moteur. Ainsi, ce système nous permet de déployer notre projet sur plusieurs système d'exploitation sans se soucier de posséder la machine supportant celui - ci.

6.2 Déployement multi-plateforme de notre jeu

Comme évoqué dans la partie «Compilation en ligne» (Cf 6), nous pouvons déployer notre jeu sur plusieurs plateformes en plus de Windows et Linux. Dans notre cahier des charges, nous avons décidé de «porter» notre jeu sur d'autre support. C'est à dire de faire les adaptations nécessaires sur notre jeu afin qu'il s'exécute aussi bien sur notre plateforme obligatoire, Windows, que sur d'autres systèmes supportés par notre moteur de jeu. Ainsi, vous avons eu la chance de pouvoir adapter «Behind Evil Corps» sur «mac OS» , le système propriétaire d'Apple Corporation, «Android» de Google et sur tous les navigateurs Internet moderne supportant une API spéciale appelée «WebGL». Ce dernier est le plus intéressant, il permet à l'aide d'un programme installé sur de nombreuses plateformes non supportés nativement sur le moteur de jeu possédant la spécification «WebGL» de fonctionner.

6.3 Demande d'utilisation de contenus réalisés pas des tiers

6.3.1 Ennemis

Comme nous l'avons étudié en cours de technique d'expression, même avec la meilleure des intentions, il est possible de faire du plagiat, sans s'en rendre compte. Alexandre a contacté chaque créateur de l'asset store du Unity individuellement, par e-mail pour leur demander leur autorisation d'utiliser leur contenu dans notre jeu. Ces e-mails, ainsi que la réponse des créateurs sont disponibles en annexes de ce rapport de projet.

6.3.2 Les musiques

La majorité de nos musiques furent produite par Teknoaxe, afin de les utiliser légalement, nous avons demandé à Matthew Huffaker de nous accorder le droit d'utilisation de sa musique dans notre jeu. Celui - ci accepta et nous a donc envoyé une licence d'utilisation. Cette licence se trouve en annexe de notre rapport de projet.

7 Expérience personnelle

7.1 Amaury LECOQ

1ere soutenance : Dans cette soutenance, j'ai été responsable du menu de départ (lorsque l'utilisateur lance le jeu). J'ai commencé par le Menu et ses butons qui affiche le nom du jeu en bas de l'écran. J'ai écrit les scripts pour l'activation des boutons vers les bonnes scènes, activation de panel, etc... J'ai fait le menu des options en mettant le nom du jeu eu haut avec une couleur différente et des nouveaux boutons d'options. Ce qui me mets à 70 au lieu de 30 d'avancement dans la section menu. Avec le menu, j'ai réalisé le site web qui est essentiel pour la communication de notre projet. J'ai crée le site web qui est fonctionnel pour, d'une part présenter l'avancée de notre projet et d'autre part le faire découvrir aux nouveaux utilisateurs.

En parallèle du site web, j'ai choisi de changer le logo car notre équipe n'était pas satisfaite de celui-ci. Ainsi, en utilisant Gimp et une image (libre de droit) de la tour Rockefeller (Empire State Building), j'ai créé le nouveau logo de notre projet que vous pouvez voir (cf annexes).

Avec ces trois choses j'ai aussi donné des idées de création dans les différents niveaux. J'ai donné quelques idées à Alexandre et Jean-Philippe pour les niveaux sachant que c'était une tache annexe pour moi.

Mon départ c'est avéré davantage difficile que ce que j'avais prévu. J'ai commencé par vouloir installer Rider sur Windows pour pouvoir faire un script sur le menu. Je ne pouvais pas l'installer à cause de Windows qui avait une version trop ancienne. Windows n'avais pas été mis à jour à default de place sur mon disque dur. Ainsi donc, j'ai perdu une journée à essayer d'installer une mise a jour qui ne voulait pas s'effectuer puis je suis passé sur Linux qui à fonctionné.

Faire le menu ma beaucoup apporté car c'était la première fois que je fessais quelque chose avec Unity. C'était aussi la première fois que j'écrivais un script sur Rider et j'ai donc appris comment fonctionnent les classes de Unity en C#. Le site web est écrit en Markdown j'ai donc appris ce langage pour l'écrire. La création du logo m'a apporté beaucoup de connaissance de logiciel de traitement d'image qui me serviront surement par la suite de ce projet. Ce début de projet ma montré que Linux est beaucoup plus pratique pour gérer ce type de projet. Il m'a aussi appris à faire des scripts ainsi que des sprites sur Unity. Enfin ce début de projet me donne envie de continuer à apprendre et à m'investir dans le jeu et l'équipe.

2eme soutenance : Cette deuxième soutenance, fut rude personnellement sous plusieurs aspects de mon travail. L'implémentation de la capacité de tir de l'intelligence artificielle plus la création du menu de pause, plus la création du manager audio ainsi que l 'implémentation des sons et la création du site internet était une charge de travail importante.

- La capacité de tir m'a posé quelques problèmes pour implémenter les armes car je devais comprendre le code d'Alexandre et de Benoît. Et ré-implémenté une partie de ce qu'ils avaient déjà fait sans tout faire planter. Mais après peu fructueux, mes efforts ont commencé a payer et j'ai enfin réussi cette mission qui m'était donnée en tant que suppléant par Alexandre.
- Le menu de pause fut assez simple à implémenter malgré quelques erreurs au niveau de la gestion du temps mais après relecture de la documentation de Unity, n'a plus posé de problèmes majeurs.
- Le manager Audio à bien fonctionné dans l'ensemble dès les premiers essais mise à par que le son n'était pas assez fort.
- La création du site internet fut compliquée à partir du moment où nous voulions ajouter une ou plusieurs images directement sur le site car les outils que nous avons utilisé ne nous le permettaient pas facilement. Ce qui était très contre intuitif sachant qu'ils étaient censé nous simplifier le travail.

Dans l'ensemble, j'ai plutôt bien réussi cette soutenance au niveau de mes

travaux personnels. Il me reste encore du travail à faire et je suis à temps voir plus en avance pour la soutenance finale ce qui va me permettre d'aider d'autres membres de l'équipes s'ils ont besoin de moi car je suis suppléant dans plusieurs domaines et entre autre j'ai gagné en souplesse. C'est ce que j'ai appris à développer lors de ce travail de groupe : la flexibilité.

Ce dernier rapport, pour moi, fut long d'une part dû au nombre de donnés à fournir pour le rapport et fastidieux dû au fait de devoir réviser tout en écrivant ce que j'ai fait. Je suis content d'avoir accompli, à l'aide de mes camarades, un jeu vidéo qui fonctionne et colle à peu près à l'idée de base. Ce fût à travers des épreuves, des défis mais aussi des énervements, que nous avons réussi en tant qu'équipe et non en tant que personne, ce jeu vidéo. Je suis fière d'avoir œuvré tel un bâtisseur aussi bien dans la fondation de notre projet que dans ses finitions les plus précises. Ce qui m'a sûrement le plus marqué est la création des menus, qui fût non des moindres une rude épreuve à travers les difficultés exposées ci-dessus mais intéressantes (sauf quand le problème ne venait pas de moi ce qui le rendait plu contraignant). À travers ce projet, j'ai appris que savoir coder n'est pas une compétence primordiale. Les compétences que je ne pensais pas si importante tel que la diplomatie, le management et l'écoute était toutes des qualités à ne pas sous estimer. J'ai appris sur moi-même et sur les gens de mon équipe pour savoir comment ils fonctionnent d'un point de vue psychologique et pragmatique.

D'un point esthétique j'ai appris à avoir une cohérence dans mes idées ainsi que dans « Unity », à savoir choisir quoi comment et pourquoi, savoir me justifier par rapport à mes choix et garder un thème et une vision par rapport à l'ambiance du jeu. Ce projet m'a appris à choisir, car au début j'étais indécis par rapport aux décisions artistiques, mais aussi à trancher et prendre la bonne décision artistique au bon moment. D'un point de vue du code, j'ai appris comment marche le C# à travers « Unity », mais d'un point de vue de pure programmation je n'ai pas appris ni beaucoup de techniques, ni fait beaucoup efforts de réflexions que je retrouve plus facilement dans les travaux pratiques que nous effectuons chaque semaine. Néanmoins, j'ai appris plein de choses sur comment utiliser « Unity » et comment faire des scripts et se simplifier la vie sur ce système. Ainsi, l'effort que j'ai fourni avec mes collègues me satisfait dans l'ensemble et j'espère que mes futurs projets se passerons de la même manière. En conclusion j'aimerai plus de communication pour mon prochain projet, il faudra que nous fassions un planning plus précis et plus simple, une connaissance meilleure sur ce que nous voudrons établir et enfin une meilleure organisation pour moins de problèmes lié au temps. Le bénéfice que je tire de cet expérience reste tout de même positif et j'espère que mes collègues pense de même.

7.2 Alexandre MOURERE

Mes missions dans notre projet concernent l'intelligence artificielle et les niveaux. Mon avancée dans ces domaines s'est révélée bien plus complexe que prévue initialement. En effet, la mise en place d'une intelligence artificielle capable de détecter un joueur m'a semblé complexe au début du projet. Cependant, en me renseignant auprès des outils mis à disposition par Unity en terme de documentation, j'ai pu mieux comprendre et diviser les tâches qu'implique un tel projet.

En effet, lorsque que mon intelligence artificielle était à son point le plus simple, elle se dirigeait vers la position du joueur, sans prendre en compte les déplacements éventuels du joueur. si l'ennemi entre en collision avec le joueur, il s'arrête. Les ennemis sont désormais capables de détecter le joueur si celui si passe dans une zone de détection. Cette zone est modifiable. Il est donc possible de placer des caméras qui seraient composées d'une zone de détection commune à tous les ennemis par exemple.Ces exemples montrent comment la division permet de régler les problèmes les plus complexes, en les réduisant à plusieurs sous problèmes, plus faciles à résoudre. Aussi, en plus de l'intelligence artificielle, j'ai également réalisé un niveau. La conception et la réalisation d'un niveau sont très différentes de celles d'une intelligence artificielle. En effet, il s'agit de deux opposés : le développement d'une intelligence artificielle consiste à programmer, à implémenter des fonctions, des méthodes, contrairement au développement d'un niveau qui ne met en œuvre que des compétences de game design. J'ai personnellement préféré la programmation d'une intelligence artificielle au développement d'un niveau.

La création d'un niveau est cependant très intéressante dans la mesure où elle met en pratique l'imagination du concepteur ainsi que son anticipation du joueur : il faut orienter le joueur vers la sortie, le faire passer au plus près des ennemis sans pour autant que celui-ci ait l'impression d'être emprisonné dans un circuit. Mon avancée bonus consiste en l'ajout de détails non prévus initialement. Par exemple, la présence de détails tels que des conteneurs permet de relier le niveau à son thème, à savoir l'usine pour cette première soutenance. Ainsi, à travers les différentes difficultés qu'il m'a été donné de rencontrer, j'ai pu améliorer ma compréhension de l'ensemble des mécanismes liés à l'intelligence artificielle. Aussi, pouvoir voir mon intelligence artificielle «prendre vie» dans le jeu est une source de satisfaction nouvelle et très enrichissante pour moi.

En effet, ma contribution à ce projet englobe les niveaux ainsi que toute l'intelligence artificielle. La progression effectuée ne s'est néanmoins pas déroulée sans obstacles. En effet, les difficultés auxquelles nous avons dû faire face se sont révélées plus complexes que ce que nous avions prévu. Cepen-

dant, en usant de persévérance, j'ai pu résoudre ces complications
En effet, l'une des principales difficultés que nous avons rencontré lors de la réalisation de notre projet est la compilation de toutes les fonctionnalités déjà implémentées. Par exemple, lors de la partie gérant le script permettant aux ennemis d'effectuer des rondes à travers la carte, il fallait bien évidemment que les ennemis stoppent leur ronde lorsque le joueur est détecté par l'ennemi lui même ou par un autre ennemi. Or, aucune variable présente dans les scripts ne remplissait cette fonctionnalité, il a donc fallu implémenté un booléen basé sur un autre script, le script en question étant celui permettant de faire en sorte que les ennemis proches du joueur le détectent. C'est donc ainsi que j'ai utilisé une astuce présente dans nos travaux pratiques : mettre une variable initialement «private» (privée), donc accessible uniquement dans le script dans lequel elle est implémentée, en public static. Ainsi, j'ai pu accéder à cette variable booléen dans d'autres scripts, notamment celui gérant les rondes. Il reste maintenant à savoir sous quelles conditions le joueur est détecté. Pour y parvenir, il est nécessaire de comprendre les moindres détails présents dans le script Detection.cs. En effet, le joueur est considéré comme détecté si :

- le joueur est présent dans la zone invisible de détection
- le joueur entre en contact avec la zone physique de détection (son corps)

Il existe quatre méthodes permettant d'affirmer si le joueur est présent ou non dans au moins une des deux options possibles. Ces quatre fonctions sont

- OnCollisionEnter : gère le cas où le joueur entre en collision avec le corps (physique) de l'ennemi
- OnCollisionExit : gère le cas où le joueur sort de la collision avec le corps de l'ennemi
- OnTriggerEnter : gère le cas où le joueur entre dans la zone de détection abstraite de l'ennemi
- OnTriggerExit : le cas où il en sort

Ainsi, on accède au booléen permettant de savoir si le joueur est détecté ou non dans tous les scripts du projet.
Cet exemple montre le type de difficultés auxquelles il a fallu faire face lors de l'implémentation des différents scripts. Cette partie à été pour moi très intéressante dans la mesure où il a été possible de mettre en application les compétences que nous avons acquises au cours des travaux pratiques, en utilisant les différentes astuces de programmation
Concernant les niveaux, même si leur conception à été une source d'inspiration moindre par rapport à l'intelligence artificielle, le fait d'être en même temps responsable de section niveaux et de l'intelligence artificielle m'a permis d'implémenter les différentes structures en même temps que le gameplay,

ce qui fournit une expérience de jeu optimale.

Ainsi, en tant que responsable général de l'intelligence artificielle du projet, j'ai implémenté les ennemis sur toutes les cartes et ait expliqué aux autres membres du groupe la façon dont l'intelligence artificielle fonctionne pour leur permettre d'implémenter également des ennemis sur les cartes. Ainsi, j'ai dû expérimenter un travail de communication au sein même du groupe pour informer l'ensemble des membres.

7.3 Benoît MALHOMME

Pour notre première soutenance, j'ai réalisé le déplacement du joueur sur la carte, la gestion des armes et de la modélisation 3D. Unity est un moteur qui, malgré sa réputation de moteur de jeu facile à appréhender, est pourtant difficile à prendre en main. Par exemple, une option qui se doit d'être visible se retrouvent souvent dans une myriade de sous menus en tout genre. L'interface de Unity peut être comparé à un cockpit d'avion, dont chacun de ces boutons pouvant mener l'avion à sa perte.

Malgré de nombreuses difficultés suites à l'apprentissage d'un logiciel de modélisation 3D qu'est Blender, j'ai pu réaliser des modèles 3D d'armes à feu et d'objets permettant de meubler nos niveaux.

Le système est gestion des matériaux de Unity possède quelques failles de conception évidentes : Lorsque le matériau d'un objet est modifié sur la carte, le moteur de jeu modifie tous les objets possédant ce même matériau. Il est alors requis de créer des matériaux identiques afin de modifier un seul paramètre tel qu'il fut évoqué précédemment.

Notre projet touche à sa fin, j'ai pu apprendre de nombreuses choses sur le fonctionnement d'un moteur de jeu très utilisé dans l'industrie du monde vidéoludique. Un moteur qui malgré son utilisation intensive par de nombreux studios se trouve entaché par des failles dans sa documentation et d'une mauvaise finition sur certains de ces outils. Aucun outil n'est parfait, mais un moteur de jeu dont des millions de dollars sont investis chaque année dans son développement, se doit de régler ces problèmes.

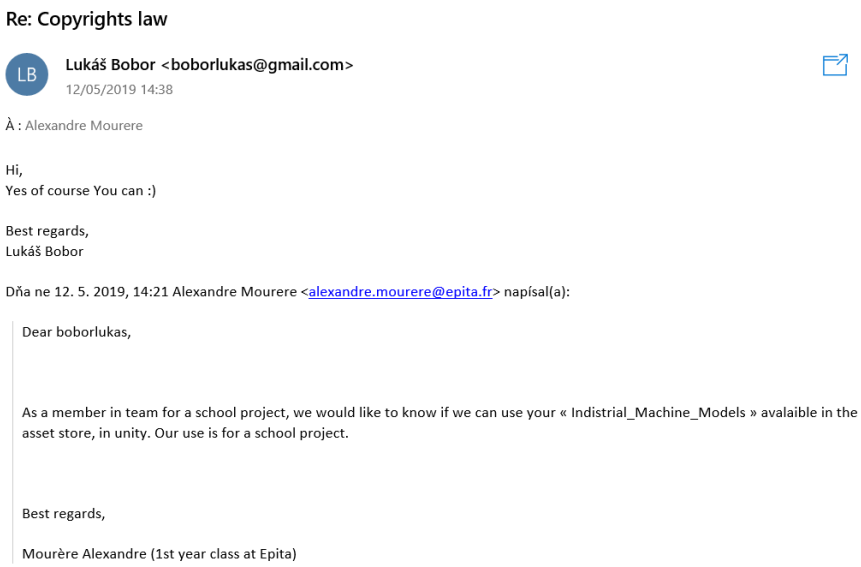
7.4 Jean-Phillipe BINGER

Je suis venu sur ce projet après que Benoît m'ait parlé de ce qu'il voulait faire pour le projet, le concept du projet dont il me parlait m'intéressait énormément et c'est pour cela que j'ai rejoins le groupe. Le projet ne m'intéressait pas que pour le concept mais aussi par rapport à ce que je pouvais en apprendre et je ne suis pas déçu de ce que j'ai appris. J'ai acquis de nombreuses connaissances vis-à-vis du système de Networking, qui est le système

de réseau en ligne, lié au jeu vidéo qui est très complexe par rapport au codage simple du mode solo. La gestion du multijoueur n'est pas la seule chose que j'ai apprise. J'ai aussi compris la gestion du temps en équipe ainsi que d'autres connaissances en informatique afin de rendre les éléments du jeu interactifs. Je déplore cependant les différents bogues rencontrés durant notre projet dont celui qui nous a supprimé une carte entière sans que l'on puisse la récupérer ce malgré des sauvegardes. Le projet dans son ensemble a été plaisant il m'a permis d'améliorer mon travail en équipe ainsi que mes connaissances personnelles en informatique.

8 Conclusion

C'est ainsi que nous avons terminé notre projet informatique de première année à l'épita. Ce projet nous est apparu comme un obstacle en premier lieu mais, avec un travail d'équipe et des compétences de plus en plus complexes. Nos compétences managériales et informatiques se sont développées : chacun est parvenu à surmonter les difficultés rencontrées sur son chemin. Ces compétences nous serons utiles tout au long de notre scolarité au sein de l'épita, ainsi en tant qu'ingénieur de demain dans les domaines de l'informatique et du numérique. Malgré les différentes complications rencontrées, il nous a été possible de les surmonter, en faisant appel à nos compétences de programmation et managériales. Le jeu est ainsi terminé et les objectifs que nous nous étions fixés atteints.



Re: Copyrights law

LB **Lukáš Bobor <boborlukas@gmail.com>**
12/05/2019 14:38

À : Alexandre Mourere

Hi,
Yes of course You can :)

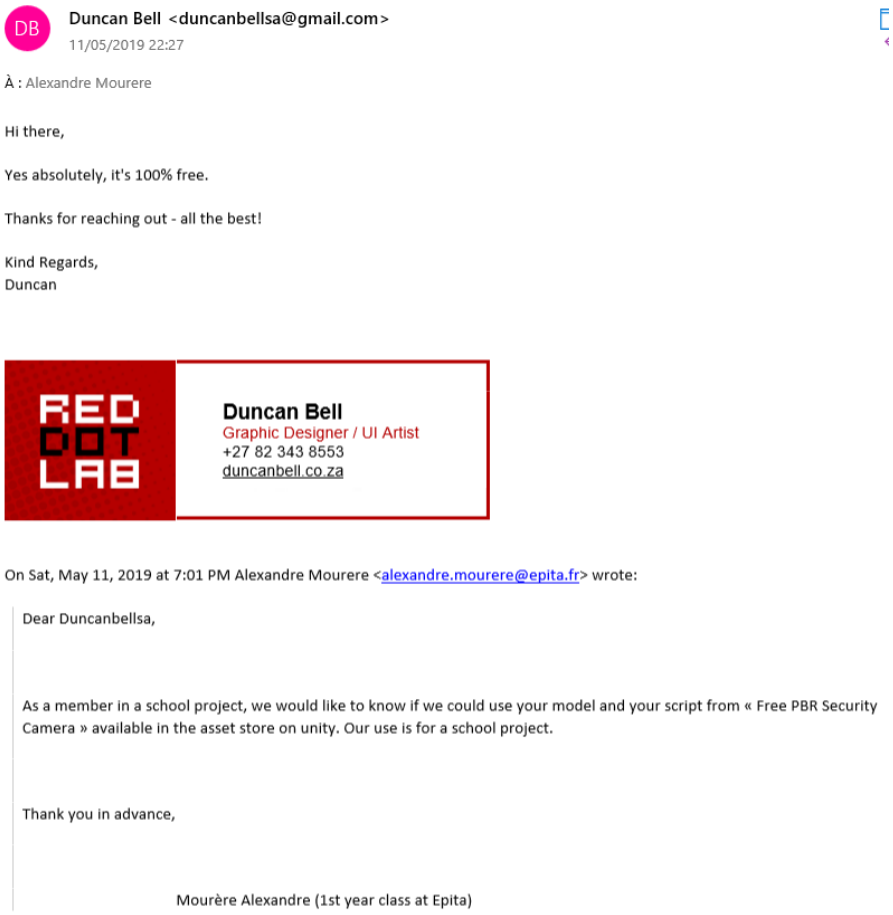
Best regards,
Lukáš Bobor

Dňa ne 12. 5. 2019, 14:21 Alexandre Mourere <alexandre.mourere@epita.fr> napísal(a):

> Dear boborlukas,
>
> As a member in team for a school project, we would like to know if we can use your « Indistrial_Machine_Models » avalaible in the asset store, in unity. Our use is for a school project.
>
> Best regards,
>
> Mourère Alexandre (1st year class at Epita)

FIGURE 1

9 Annexes



DB **Duncan Bell <duncanbellsa@gmail.com>**
11/05/2019 22:27

À : Alexandre Mourere

Hi there,

Yes absolutely, it's 100% free.

Thanks for reaching out - all the best!

Kind Regards,
Duncan

RED DOT LAB

Duncan Bell
Graphic Designer / UI Artist
+27 82 343 8553
duncanbell.co.za

On Sat, May 11, 2019 at 7:01 PM Alexandre Mourere <alexandre.mourere@epita.fr> wrote:

> Dear Duncanbellsa,
>
> As a member in a school project, we would like to know if we could use your model and your script from « Free PBR Security Camera » available in the asset store on unity. Our use is for a school project.
>
> Thank you in advance,
>
> Mourère Alexandre (1st year class at Epita)

FIGURE 2

RR Robert Ramsay <me@robertramsay.co.uk>
11/05/2019 19:01

À : Alexandre Mourere

Hi,

Yes you can use this pack for learning, hobby, project, commercial or other. It has not got any restrictions.
Enjoy!

Thanks,
Robert

Freelance

Robert Ramsay
3DFolio
Mob +44 7742433302

On 11 May 2019 at 17:55:34 +01:00, Alexandre Mourere <alexandre.mourere@epita.fr> wrote:

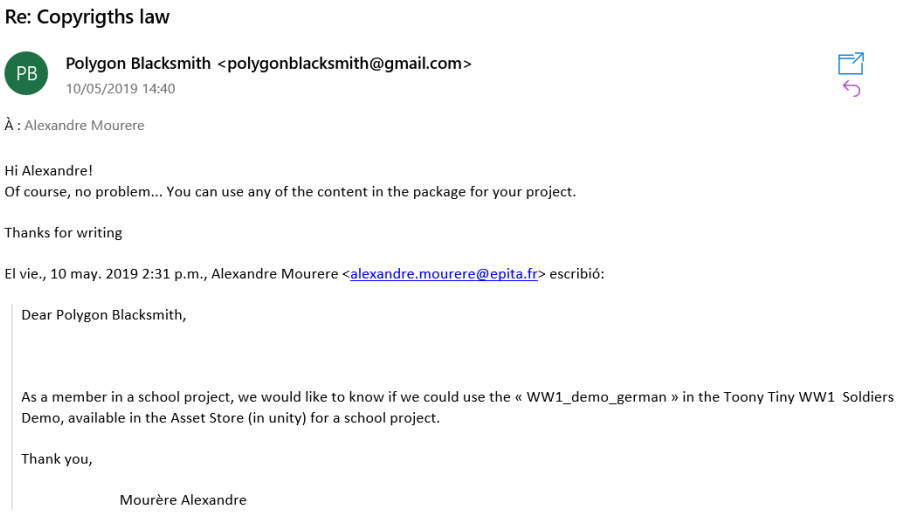
Dear RRFREELANCE,

As a member in a school project, I would like to know if we could use your « Low Poly Office Props – LITE » model available on the Asset Store on unity. Our use is for a school project.

Thank you,

Mourère Alexandre (1st year class at EPITA)

FIGURE 3



Re: Copyrigths law

PB **Polygon Blacksmith <polygonblacksmith@gmail.com>**
10/05/2019 14:40

À : Alexandre Mourere

Hi Alexandre!
Of course, no problem... You can use any of the content in the package for your project.

Thanks for writing

El vie., 10 may. 2019 2:31 p.m., Alexandre Mourere <alexandre.mourere@epita.fr> escribió:

> Dear Polygon Blacksmith,
>
> As a member in a school project, we would like to know if we could use the « WW1_demo_german » in the Toony Tiny WW1 Soldiers Demo, available in the Asset Store (in unity) for a school project.
>
> Thank you,
>
> Mourère Alexandre

FIGURE 4

To Whom it may concern.

I, Matthew Huffaker, (http://www.teknoaxe.com),(http://www.youtube.com/user/teknoaxe),
(teknoaxe@gmail.com) give permission to
Benoit Malhomme,
channel name: PastaOverflow,
email: benoit.malhomme@epita.fr,
permission to use my music, which is hosted on http://www.teknoaxe.com, and
represented on the youtube channel http://www.youtube.com/user/teknoaxe under the
terms of CC (Creative Commons) 4.0 license.

Permissions extend to commercial and non-commercial purposes. Videos with TeknoAXE
music can be monetized and/or used for commercial purposes. Streams with TeknoAXE
music can be monetized and/or used for commercial purposes. Games are allowed to
feature TeknoAXE music so long as it does not infringe on others' rights of usage.
TeknoAXE music can be modified/edited in length and pitch to fit individual needs.

Sufficient citation can include links to the youtube channel,
http://www.youtube.com/user/teknoaxe, the youtube video from
http://www.youtube.com/user/teknoaxe that represents the track used in Benoit Malhomme's work
a direct link to http://www.teknoaxe.com, and/or the webpage on http://www.teknoaxe.com
that hosts the track used Benoit Malhomme's work.

Signed by,
Matthew Huffaker, TEKNOAXE
Thursday 16th of May 2019 04:56:01 AM

FIGURE 5 – Autorisation d'utilisation de la musique dans notre jeu