Soutenance

PastaOverflow

25 avril 2019

1 Introduction

Les progrès que nous avons effectué dans les domaines de l'intelligence artificielle, de la modélisation en trois dimensions des armes, des niveaux et des menus fut pour nous une source d'encouragement face aux difficultés que nous avons rencontré lors de notre progression dans le projet. Ces difficultés nous ont permises de progresser d'un point de vue technique. Nous avons du travailler en équipe pour parvenir à les pallier. Dans une première partie, nous étudierons les avancements effectués depuis la première soutenance selon chaque membre du groupe. Ensuite, nous détaillerons les avances et les retards accumulés lors de cette soutenance. Finalement, nous verrons les avancées prévues pour la soutenance finale. Nous conclurons par une analyse individuelle du projet.

2 Avancement par Parties

2.1 Intelligence artificielle

2.1.1 Pathfinding

Alexandre :

Lors de la première soutenance, les difficultés rencontrées étaient liées au fait que je mesurait la difficulté d'implémenter une intelligence artificielle rudimentaire. Cependant, nous avons rapidement surmonté ces complications aisément du fait de leur complexité relativement faible. En revanche, lors de cette soutenance, j'ai du faire face à de nouveaux enjeux, dont celui de réussir à rendre l'intelligence artificielle complète, c'est à dire, à parvenir à assembler toutes les briques que j'ai implémentées. En effet, lors de la complexification de l'intelligence artificielle, je voulais que les ennemis puissent alerter les autres ennemis lorsque le joueur est détecté, ce qui est le fondement d'un jeu d'infiltration. Il fallait aussi que les ennemis puissent faire des rondes pour permettre à la fois au joueur de s'infiltrer entre les ennemis et à la fois de devoir anticiper les déplacements de l'intelligence artificielle. Ces deux fonctionnalités doivent bien évidemment pouvoir fonctionner ensemble mais aussi avec le système de détection déjà en place. La difficulté de ces fonctionnalités est relativement difficile. Par exemple, pour la possibilité d'effectuer des tours d'un point A à un point B dans la carte repose sur un système de Waypoint, cf annexe correspondante [ANNEXE]. Ce système est basé sur des points appelés waypoints que l'on ajoute à un tableau de waypoints. Ainsi, la

manipulation de ce tableau de waypoints est l'élément principal des rondes. Nous allons créer une variable ayant le type entier qui permet de parcourir le tableau d'élément, que nous nommons IndexWaypoints. Cette valeur est utilisée de la manière suivant : on récupère le waypoint contenu dans la case correspondant à la valeur d'IndexWaypoints. Nous demandons, à l'aide des vectorPosition de déplacer l'ennemi vers la position du waypoint. Ensuite, nous incrémentons IndexWaypoints jusqu'à atteindre la fin du tableau d'éléments. Lorsque cela se produit, j'ai choisit de forcer l'ennemi à refaire sa ronde, dans le sens opposé, pour y parvenir, nous décrémentons la valeur de IndexWaypoints. Enfin, lorsque nous arrivons au premier élément du tableau, on recommence le programme. Il faut néanmoins être vigilant car il ne faut pas que l'ennemi continue à faire des rondes si le joueur est détecté. Il faut donc mettre le programme précédent dans une structure conditionnelle exigeant que le joueur ne soit pas détecté. Pour y parvenir, nous devons utiliser le programme de détection que j'ai implémenté pour la première soutenance, en ajoutant une variable publique et statique afin d'y accéder dans un autre script, ici le script permettant d'effectuer les rondes. Nous avons donc notre variable publique (de type booléen) permettant de dire si le joueur est détecté ou non. Cependant, il faut savoir, dans le script de détection, quand le joueur est détecté. En effet, celui-ci gère plusieurs cas : le cas où le joueur est au contact de l'ennemi, le cas où il en sort, le cas où le joueur est situé dans la zone de détection et enfin le cas où il en sort. Dans chacun de ces cas, le booléen public permettant de savoir si le joueur est détecté est modifié, lorsque le joueur entre en contact physique avec l'ennemi ou lorsqu'il entre dans la zone de détection. Dans ces deux options, on l'initialise à true, sinon à false. Ainsi, nous avons une intelligence artificielle capable d'effectuer des rondes dans la carte, tant que le joueur n'est pas détecté, auquel cas il attaque le joueur puis reprend sa ronde au bon endroit (le dernier auquel il était). Ainsi, la possibilité de faire des rondes nécessite une excellente compréhension du code précédemment implémenté et souligne aussi le fait que les programmes sont interdépendants. La deuxième partie du pathfinding correspond au chemin que choisit l'intelligence artificielle pour attaquer le joueur. Le premier script détection constitue déjà une première approche du pathfinding, avec des outils relativement primaires. Cependant, les outils utilisés dans ce script sont bien plus complexes, nous utilisons une fonctionnalité appelée NavMesh, qui permet de scanner l'ensemble de la carte pour détecter le meilleur chemin (le plus court) entre un point A et un point B, avec le point A correspondant à la position actuelle de l'ennemi et le point B représentant la position du joueur. Cet outil NavMesh est d'une puissance remarquable dans la mesure où le calcul du meilleur chemin est relativement simplifié. Cependant, il faut implémenter cette fonctionnalité en accord avec celles déjà présentes. En ef-

fet, il faut que cette fonctionnalité ne s'applique que si le joueur est détecté par au moins un ennemi. Ainsi, nous pouvons ré-utiliser le booléen précédemment utilisé (dans la partie permettant aux ennemis de faire des rondes). Si le joueur est détecté, alors les ennemis doivent rejoindre la position du joueur lorsqu'il a été détecté. Ceci est un choix de gameplay car nous considérons que l'ennemi appelle ses collègues pour l'aider, et que ceux-ci se retrouvent à la position indiqué. Cette fonctionnalité permet au joueur de fuir si besoin. Cependant, il faut tout de même que les ennemis, une fois que le joueur n'est plus détecté puissent reprendre leur ronde. Ainsi, au sein de cette fonction nous mettons une variable permettant de définir si les ennemis doivent effectuer le déplacement ou non, dans ce cas, ils continuent leur ronde, jusqu'à ce que le joueur soit à nouveau détecté. Enfin, il faut que cette partie puisse s'intégrer correctement aux autres fonctionnalités, ce qui a été permis par l'implémentation de différentes variables liant les trois parties entre-elles. La difficulté majeure rencontrée lors de l'implémentation de l'intelligence artificielle à été de tout mettre en accord. Un exemple de problème rencontré lors de l'implémentation est qu'initialement, les ennemis n'avançaient pas. Un débogage s'est donc révélé nécessaire, à l'aide du mot clef public static, nous pouvons accéder au tableau waypoints du script rondes, j'ai constaté qu'il n'était pas vide, le problème ne se situait donc pas ici, il est donc dans le script même de l'intelligence artificielle. Pour le trouver, on utilise une méthode appelée Debug.Log() qui permet d'afficher un message dans la console, un équivalent du Console.WriteLine() de C#. Nous affichons donc un message quand l'ennemi veut se diriger vers un waypoint. On constate que l'ennemi veut effectivement se déplacer vers un waypoint, le problème est donc précisément localisé, il s'agit de la condition permettant à l'intelligence artificielle de se déplacer vers un waypoint qui empêchait tout déplacement. Ainsi, en modifiant la variable déterminant si le joueur est détecté ou non, j'ai réussi a permettre le déplacement de l'ennemi vers son waypoint. Cet exemple montre à quel point les parties sont interdépendantes et peuvent entrer en conflit, ce qui génère des erreurs. La partie concernant le pathfinding de l'intelligence artificielle s'est très largement complexifiée. Néanmoins, ces difficultés peuvent et ont été surmontées à l'aide des habitudes acquises lors des travaux pratiques, en plus de la compréhension globale et précise de l'ensemble du script du jeu. Les compétences et les connaissances acquises lors de ces TP m'ont permis de résoudre un certain nombres de ces problèmes rencontrés, tels que l'astuce de mettre une variable du script de la détection en

```
public static
```

pour en permettre l'utilisation dans d'autres scripts, par exemple.

2.1.2 Capacité de faire feu

Alexandre, Amaury :

Nous avons annoncé, lors de la première soutenance que les ennemis seraient capables, en plus de pouvoir faire des rondes, de tirer sur le joueur lorsque celui-ci est à portée de tir et est détecté par un ennemi. Nous avons réussi cette partie. Néanmoins, aux vues des difficultés rencontrées lors de l'implémentation des rondes, j'ai décidé de laisser Amaury implémenter cette fonctionnalité, en tant que suppléant à l'intelligence artificielle. En tant que suppléant de l'intelligence artificielle, le début à été compliqué pour comprendre et utiliser les outils que Alexandre avait implémenté. Effectivement, après une période d'adaptation, j'ai commencé à utiliser les fonctions qu'Alexandre avait écrit et utilisé plusieurs variable de la détection pour savoir si l'on doit tirer sur le joueur. L'intelligence artificielle vérifie que le joueur et dans la direction de celle ci et qu'elle a détecté celui ci avant de tirer. Ainsi si elle peut tirer, alors j'ai ré implémenté les fonctions, méthodes et classes de benoit pour tirer sans les munitions ni le changement d'armes pour l'intelligence artificielle. L'intelligence artificielle tire et attend avant de pouvoir retirer. Comme par exemple la variable canshoot qui est un booléen qui dit à l'intelligence artificielle si elle à le droit de tirer ou pas. Le tire consiste à créer un rayon sur la cible et vérifier que celle-ci et bien un joueur. Elle doit atrndre grace à un threat.sleep qui force l'intelligence artificelle des ennemis à attendre le temps définis parl'arme. Le cas échéant, on appelle la classe Playerlife.Hit pour enlever la vie correspondant aux damages de l'arme au player. Une diffculté majeur qui m'as pris un certain temps à résoudre fut lors de la détection du joueur par la raycast (lancé de rayon), c'est à dire que on doit regarder si le tag du de l'objet touché doit être "Player" car le tag.name de l'objet player est "Player". J'avais écrit dans le code le mot "PLayer", car je voulais surement aller vite, qui est très proche mais qui n'est pas pareil et difficilement différentiable. Je ne comprenais donc pas pourquoi le Player n'était pas touché alors qu'il était largement en face de celui-ci. Après plusieurs relecture et débogage intense, j'ai enfin trouvé l'erreur et le code est passé d'incorrect à juste et utilisable. Ainsi nous avons réussi à implémenter le tir de l'intelligence artificielle qui pet désormais tuer le joueur. Pour gérer la vie du joueur, j'ai été contraint de créer la classe player life avec une fonction hit qui décrémente la vie du joueur en fonction des damages reçus. Ainsi dans cette classe il y a une variable playerLife qui correspond a la vie du Joueur. Si la vie du Joueur est inférieur ou égal à 0, alors le player est considéré comme mort et nous lancerons une scène de fin qui sera implémenté dans la prochaine soutenance.

2.2 Niveaux

2.2.1 Architecture

Jean-Philippe :

Pour la première partie de la map nous avons décidé de se baser sur la structure d'un hangar dont le but est d'atteindre l'autre bout du hangar pour terminer cette partie du niveau. Des armes sont disposées le long du le chemin afin d'affronter les ennemis rencontrer si cela s'avère nécessaire. Des ennemis sont présents sur la map empêchant de traverser la map sans difficulté. Des conteneurs, des caisse et des barils sont disposé permettant d'avoir une couverture mais aussi d'empêcher le joueur d'aller à la fin du niveau avec une simple ligne droite. Une passerelle est disposé faisant le tour de la carte est de contourner un obstacle. La disposition des conteneurs, des caisses et des barils est faite de manière à permettre au joueur de ne pas avoir un décor trop redondant.

Alexandre

Lors de la première soutenance, nous avons expliqué que nous détaillerons les deux cartes implémentées, en plus de l'ajout de trois autres cartes, une se déroulant dans l'univers industriel, non texturée et non détaillée, deux autres cartes se déroulant dans le thème des bureaux d'une entreprise. Je me suis personnellement occupé de l'implémentation des cartes se déroulant dans le thème des bureaux d'une entreprise. La première carte correspond à un labyrinthe dans un bureau type open-space, le joueur doit, en esquivant les ennemis faisant des rondes dans les couloirs du bureau, s'infiltrer, trouver le boss du niveau pour l'éliminer. Certains ennemis sont disposés de sorte qu'il surprendra le joueur. Ainsi, le joueur devra anticiper les déplacements ennemis prévoir plusieurs stratégies s'il se fait repérer.
La deuxième carte à été développée en deux parties (cf image 1) : la première partie est la suite de la première carte, le joueur arrive dans une salle contenant un certain nombre de bureaux, dans laquelle la présence d'ennemis sera maximale. Cette partie sera plus nerveuse que précédemment, le joueur devra nécessairement faire face aux ennemis qui lui tireront dessus. Des armes seront placées dans le but d'aider le joueur à survivre à cette partie. S'il survit, il accèdera à la phase finale de cette carte. Cette phase représente un garage dans lequel il devra s'infiltrer entre les voitures garées pour exécuter le boss de fin de carte. Une fois ce dernier ennemi éliminé, le joueur accède à la sortie de la carte et accède à la dernière map du monde bureaux.

2.2.2 Détails

Jean-Philippe :

Pour les détails J'ai ajouter diverse taille de caisses ainsi que des barils. Des conteneurs sont parfois ouvert avec un différent nombre de caisses à l'intérieur des portes de conteneurs ouverte différemment. Les différents objet sont positionner différemment pour varier le décor.

Alexandre

A l'aide de la modélisation en trois dimensions implémentées, nous possédons des voitures, des écrans d'ordinateurs et des unités centrea

Dans le cahier des charges, il a été précisé que nous n'ajouterions pas de détails pour nos niveaux. Cependant, nous avons pris de l'avance en ajoutant des conteneurs, des caisses et des lumières.(cf image 2).

2.3 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 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 :

```
if (hit.transform.tag == "Enemy")
{
hit.transform.gameObject.GetComponent<Enemy>().Hit(playerWeapons[currentW
}
```

2.4 Modélisation 3D

2.4.1 Armes

2.4.2 Ennemis

2.5 Aspect Sonore

2.5.1 Effets sonores

Benoit :

J'ai enregistré, à l'aide d'un micro studio et d'un ordinateur, des sons d'une arme à feu dans un centre d'armes à feu à Strasbourg. J'ai tiré, en étant encadré par un professionnel de tir, dans un stand de 25 mètres. M'étant aperçu du faible nombre de sons, j'ai pris d'enregistrer des sons d'une arme à air comprimé.

2.5.2 Musique

Amaury :

L'écriture, la composition et l'enregistrement d'une musique d'un jeu vidéo nécessite du matériel dont la mise en place s'est révélée plus complexe que ce que nous avions prévu lors de la première soutenance de notre projet, c'est pourquoi, les musiques étant composées dans ma tête ne sont pas encore enregistré mais ça ne saurait tarder. En revanche, on peut noter une avance non négligeable qui figure parmi les options les plus sophistiquées qu'est le manager audio.

Le Manager Audio est constitué de deux classes. Une première classe Sound qui est une classe de variables que l'on peut modifier directement sur Unity qui comprend : le nom de la musique pour la retrouvé lors de la recherche pour lancer la musique, le chemin pour y accéder qui servira à créer la source audio pour lancer le fichier dans le jeu, le volume de la musique ou le son dans le jeu pour régler en fontion du fichier source, le pitch de la musique ou du son dans le jeu qui sert à la même utilisation que précédent, une variable booléenne loop (boucle en anglais) qui nous permet de savoir si la musique doit tourner en boucle ou pas (le thème principal par exemple ou autre) et une variable "cachée" dans unity qui servira pour contenir la variable source qui nous permettra de tout modifier et de lancer le son grace au variables ci dessus que nous allons lier ensemble dans la classe suivante : AudioManager.

Dans cette classe, on retrouve une liste statique de son (de la classe Sound) appelé "sounds". Ensuite nous avons une variable instance qui est de type

AudioManager qui sert à éviter les copies dans les différentes scènes de notre jeu de notre Audiomanager. Car on ne veut pas que la musiques se coupes quand nous passons d'une scène à l'autre, c'est pourquoi Nous utilisons dans Awake (qui commence avant le start) une methode appellé DontDestroyOnLoad qui ne détruit pas l'objet d'une scène à l'autre. ainsi avant de lancer cette commande nous vérifions que l'instance est bien null et sinon nous détruisons l'objet courant (l'audio manager créé dans la scène). Ensuite on associe chaque sources de chaque musique a un objet AudioSource qui gère les sources audio et pour chaque paramètre que nous avons, nous lui associons son paramètre Audio source (Volume, clip pitch, loop). Ensuite Nous avons une fonction PLay dans cette classe audiomanager qui prend en paramètre un nom en string, qui le cherche dans la liste des songs et qui le lance sur l'audiosource. Si le son ou la musique n'est pas trouvé, la fonction renvoie un warning. Pur lancer une musique dans n'importe quelle classe, nous pouvons écrire la ligne suivante :

```
public void StartGame()
{
        FindObjectOfType<AudioManager>().Play("nom du
           son a lancer");
}
```

La fonction cherche l'objet Audiomanager et lance la fonction play avec le nom de la musique correspondant.

2.6 Autre

2.6.1 Multijoueur

Jean-Philippe :

Le multijoueur est encore en cours de développement et n'est donc pas encore implémenté dans le jeu. Certains objets capitaux n'ont pas pu être adaptés pour le multijoueur puisque ces derniers étaient encore en développement. Ces objets furent développés en priorité pour le mode solo du jeu. Le développement du multijoueur en est au stade de devoir reconnaitre les objets propre à chaque client.

2.6.2 Menu

Amaury :

Le Menu se divise en deux partie. La première partie est le panel du menu

avec l'image du logo en noir et blanc le fond et les boutons jouer (Play), options (Options) et quitter (Quit). Ainsi j'ai fait plusieurs scripts pour chaque boutons.

Le bouton Play lance la fonction StartGame() qui charge la scène du jeu principale appelée startSceneName.

```
public void StartGame()
{
        SceneManager.LoadScene(startSceneName);
}
```

Le bouton Options lance la fonction Options() qui active le panel des options (devant le menu principale).

```
public void Options()
{
        OptionsPanel.SetActive(true);
}
```

Le bouton Quit lance la fonction Quit() qui ferme le jeu.

```
public void Quit()
{
        Application.Quit();
}
```

Puis j'ai fait le menu des options sans les options (car nous n'avons pas encore implémenté tout ce qui est nécessaire pour le faire). Le menu des options se compose de deux images de pistolets qui rappelle le coté action du jeu, de trois boutons pour le menu et d'un bouton de fermeture du menu des options.

Le bouton de fermeture est implémenté par un script avec deux fonctions. La première est le Start() qui initialise la valeur du panel lui même.

```
private void Start()
{
        OptionsP =
           OptionsP.GetComponent<MainMenu>().OptionsPanel;
}
```

La fonction QuitOptionsMenu() est appelée quand on clique sur le bouton de fermeture du menu d'options qui rend le panel des options inactif.

```
public void QuitOptionMenu()
{
```

```
        OptionsP.SetActive(false);
}
```

Vous pouvez voir le rendu de ces deux menus en annexes de ce rapport.

2.6.3 Site Internet

Amaury :

Le site internet à été implanté avec hugo (un interpréteur de markdown) et voici le résultat du post que j'ai créé pour la première soutenance :

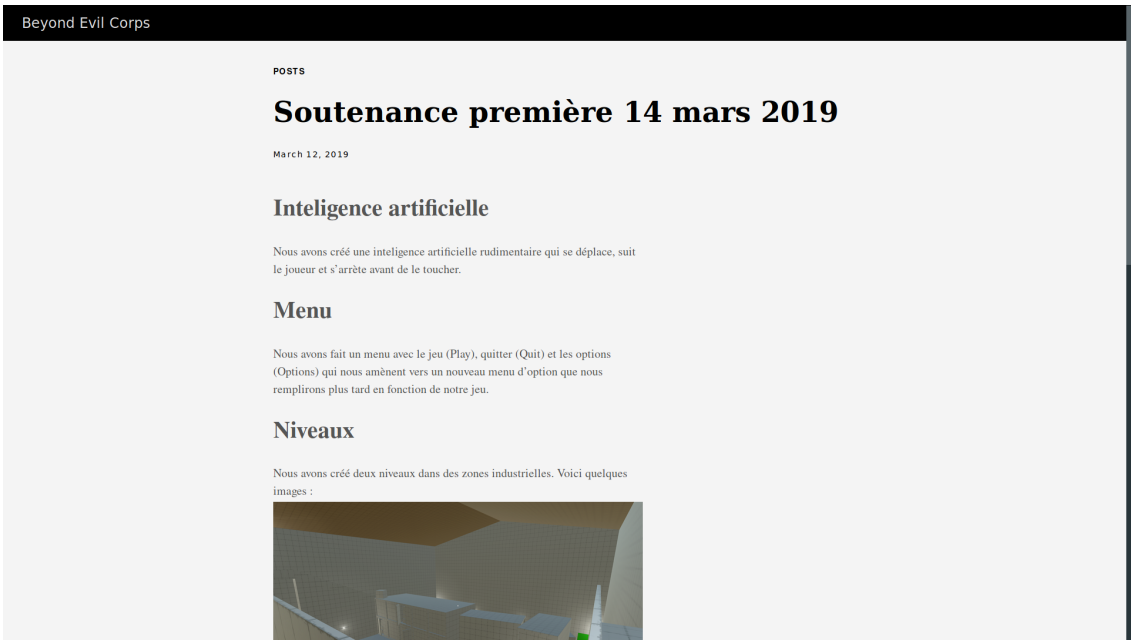

FIGURE 1 – Le Post sur la Première Soutenance

3 Avances et Retards

3.1 Intelligence artificielle

3.1.1 Avances

Alexandre :

— Détection du joueur simplifiée : il est possible d'adapter la zone de détection des ennemis

— Pathfinding :
  — les ennemis se déplacent vers le joueur, de face (donc se capables de se tourner) en s'arrêtant devant lui
  — Si le joueur se déplace hors de la zone de détection des ennemis, ils arrêtent de le poursuivre

3.1.2 Retard

Alexandre :

Aucun

3.2 Niveaux

3.2.1 Avances

Alexandre et Jean-Philippe

Ajout de détails :
— conteneurs
— caisses
— lumières colorées

3.2.2 Retard

Alexandre et Jean-Philippe
Aucun

3.3 Gameplay

3.3.1 Avances

Benoit
J'ai implémenté deux armes au lieu d'une.

3.3.2 Retard

Benoit
Le bouton de course à mal été implémenté

3.4 Modélisation 3D

3.4.1 Avances

Benoit
Aucunes

3.4.2 Retard

Benoit
Tout est modélisé de façon rudimentaire. Faute de connaissance de Blender. La tache à été sous-estimé.

3.5 Aspect Sonore

3.5.1 Avances

Amaury et benoit
Aucunes

3.5.2 Retard

Amaury et benoit
Aucun

3.6 Autre

3.6.1 Avances

Amaury
Nous sommes en avance sur le menu car nous voulions faire 2 boutons mais nous en avons 3 et un menu d'options en plus.

3.6.2 Retard

Jean-Philippe
Retard sur le multi-joueurs car impossibilité de l'implémenter à ce stade.

4 Avancements Prévus à Venir

4.1 Intelligence artificielle

4.1.1 Pathfinding

Alexandre :

Nous voudrions également faire en sorte que les ennemis fassent des rondes dans les couloirs. Pour y parvenir, j'implémenterai une fonction, basée sur un booléen (qui déterminerait si le joueur est détecté ou non par un ennemi) qui bloquerait les déplacements sur des points précis (pour les allers/retours). Cependant, si le joueur est détecté par un ennemi, la méthode ne sera plus appelée et les ennemis se déplaceront vers le joueur (en tirant, dont le script aura d'abord été implémenté).

4.1.2 Détection du Joueur a Distance

Alexandre :

Pour la deuxième soutenance, nous voudrions que les ennemis possèdent l'action de tirer : ils pourront techniquement tirer, il ne manquera plus que le script permettant que les ennemis tirent, ce qui est situé dans la section "gestion des armes". Mon objectif est d'augmenter l'interaction avec le joueur : pour cette première soutenance, ils se contentent de fixer le joueur. Ainsi, les ennemis auront un objet associé qui sera l'arme (dont le script appellera celui implémenté par le responsable de la gestion des armes). Pour que l'ennemi puisse tirer, il faudra que le joueur se situe dans sa zone de détection, qu'il n'y ait pas de mur (ou d'ennemis) entre lui et le joueur. Une méthode faisant appel à au moins deux booléens assurant ces conditions est nécessaire. Si et seulement si ces conditions sont respectées, je ferai appel (dans une autre méthode) au script utilisé dans la perte des points de vie. La cadence de tire sera déterminée par le responsable de la section gestion des armes. Aussi, à l'aide des tags des objets du jeu, je ferai en sorte que les ennemis ne se tirent pas dessus.

4.2 Niveaux

Alexandre et Jean-Philippe

Lors de la deuxième soutenance, nous aurons ajouté 3 cartes : Une dans l'univers de l'usine et les deux autres dans le thème des bureaux. La première carte se déroulera dans une usine, elle sera très détaillée et non texturée. Les deux autres cartes se dérouleront dans des bureaux, elles ne seront pas

détaillées et non texturées. Aussi, nous aurons détaillé les premières cartes et commencé à ajouter des textures.

4.3 Gameplay

4.3.1 Déplacement du Joueur

Benoit

Pour la prochaine soutenace, je prévois d'ajouter une animation de marche et de course pour le joueur en modifiant la hauteur de la caméra. La majorité des armes sont déjà implémentées. Il reste à ajouter le fusil d'assaut. Aussi, il faut que j'ajoute un mode permettant aux ennemis de tirer afin que le responsable intelligence artificielle puisse utiliser le script.

4.4 Modélisation 3D

Benoit

Les ennemis doivent encore être modélisés, leur skin sera celui indiqué dans le cahier des charges. Il existera trois types d'ennemis : le léger (en costume), le normal (avec un gilet par-balles) et le gros (en kevlar). Deplus, il me faut encore modéliser le joueur afin de lui donner une apparence concrête et définitive (évidemment en lien avec le thème de notre jeu). Les armes seront modélisées, elles auront plus réaliste et possèderont des animations (tirs et rechargement).

4.5 Aspect Sonore

4.5.1 Effets sonores

Benoit :

Les coups de feu des armes sont enregistrés mais restent brut. En effet, la majeure partie nécéssite un traitement audio afin de les rendre plus percutants et plus agréable à écouter. C'est à dire, faire paraître l'arme puissante sans provoquer un mal de tête.

4.5.2 Musique

Amaury :

Composition d'une ou plusieurs musique en fonction de mon inspiration et implémentation des musiques libre de droit.

4.6 Autre

4.6.1 Multijoueur

Jean-Philippe :

Pour la seconde soutenance le multijoueur sera implémenté de manière à ce que 2 joueurs puissent se trouver en même temps sur une map conçu spécialement pour ce mode de jeu. Le choix pour le mode multijoueur sera donc aussi ajouté. La gestion des joueurs, des armes et des ennemies sera ajoutée. Nous comptons faire en sorte que notre mode multi-joueurs puisse s'intégrer aussi bien dans la vision que nous avons de notre projet informatique que dans sa version la plus concrète et finale.

4.6.2 Menu

Amaury :

Faire un sous menu entre multi-joueur et le mode solo. ajout d'un menu interne au jeu avec la touche échap.

5 Expérience personnelle

5.1 Benoît

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 apprhender, 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.

5.2 Alexandre

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.

5.3 Amaury

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.

5.4 Jean-Philippe

Ma partie consiste en la création de la première partie de la carte du premier niveau. Cette carte se trouve dans un univers industriel, elle sera alors un hangar de stockage entièrement fermé. La carte s'avère difficile à créer avec les outils de base de unity mais s'en est vus facilitée lorsque l'on commence à maitriser l'outil probuilder. Cet outil permet ainsi d'ajouter facilement des objets plus complexes que les objets basiques fournit par unity. J'ai du apprendre à utiliser probuilder en essayant de faire différents objets basiques tels que des murs ou des caisses de différentes tailles, me permettant

ainsi de faire des structures plus complexescomme des structures trouées et des ensembles d'objet permettant de faire un conteneur. Il permet aussi de modifier n'importe quel objet ayant déjà été placé sur la carte. L'idée du hangar m'inspira pour faire une sorte de chemin défini par les caisses et les conteneurs rangés de manière organisé vers le fond du hangar, devenant de plus en plus désorganisés vers les portes d'arrivées des marchandises. Le point de départ ayant été défini comme une porte de sortie de secours qui est alors placée plus vers l'avant du bâtiment et le point d'arrivée une autre porte de sortie se trouvant à l'opposé du hangar.

L'idée de base du niveau est de pouvoir se cacher à l'intérieur de certains conteneurs laissés ouverts. La création de ce niveau à été difficile dû à la corruption des données de la carte dès la première création de la carte empêchant ainsi de sauvegarder cette dernière faisant ainsi perdre une grande quantité de travail. La dernière difficulté vis à vis de la carte fut l'ajout des lumière qui avaient un défaut de génération, laissant ainsi de temps à autre des lampes disparaître lors de changement d'angle de la caméra du joueur.

La création de carte n'étant pas mon unique rôle, je me suis lancé dans la gestion du multijoueur. L'implémentation de ce dernier est encore en développement puisque pour faire cela il est nécessaire de modifier les scripts du "player" ainsi que celui des armes et des ennemis qu'il faut adapter au multijoueur car ces scripts sont penser au départ pour le jeu solo mais étant corriger régulièrement la modification de ces derniers nécessite d'être sûr de ne plus modifier le script après l'adaptation. Je me suis alors renseigner sur la gestion du mode multijoueur permettant de l'ajouter dès que l'adaptation sera possible.

Ces recherches m'ont permis de comprendre les bases d'un système de room se trouvant dans un lobby et la gestion des clients de chaque room avec, comme finalité, de mettre en place, d'installer un mode multi-joueurs efficace, intéressant, optimisé et en lien avec notre version propre et unique de notre projet, notre jeu, en groupe en informatique.

6 Conclusion

De notre idée, Behind Evil Corp, est née un intérêt grandissant pour le développement d'un projet informatique en groupe. La confrontation permanente de nos idées à la réalité est parfois difficile et plus longue que prévu. Notamment des problèmes inopinés. Ces problèmes étant parfois très frustrant, il est néanmoins très satisfaisant de les résoudre. Certain au travers de la programmation on put avoir une premier approche de l'intelligence arti-

ficielle. Nous avons dans l'ensemble appris à utiliser Unity, même si parfois son utilisation c'est avéré hasardeuse !

7 Annexes

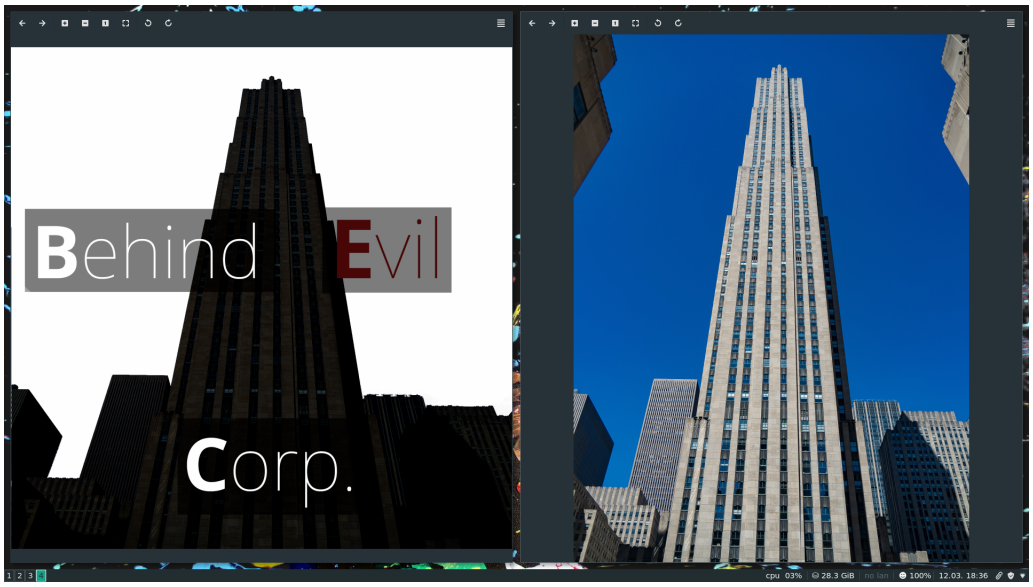

FIGURE 2 – Notre logo Avant-Après

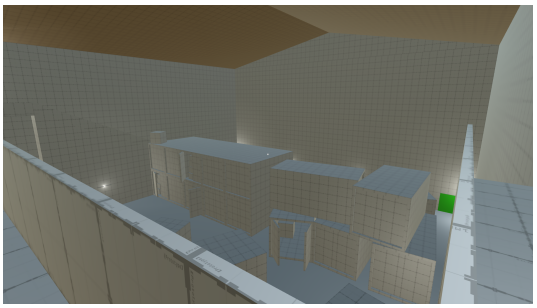

FIGURE 3 – Partie 1 du Niveau 1 in Game a partir de la passerelle

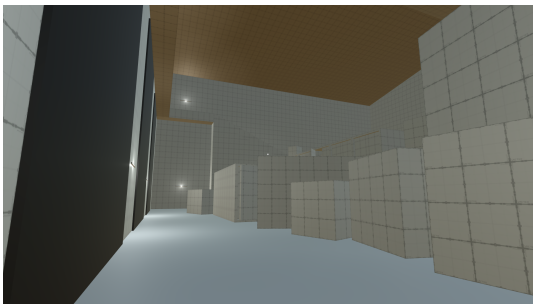

FIGURE 4 – Partie 1 du Niveau 1 in game Départ

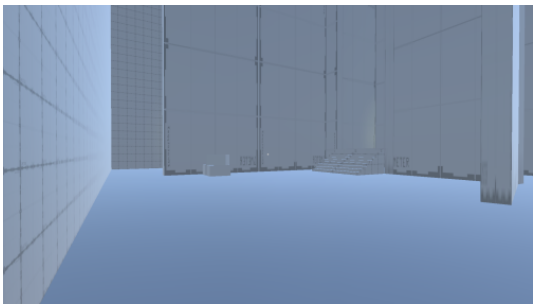

FIGURE 5 – Level 2 image 1

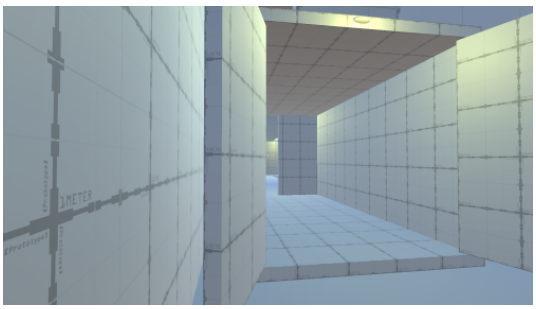

FIGURE 6 – Level 2 image 2

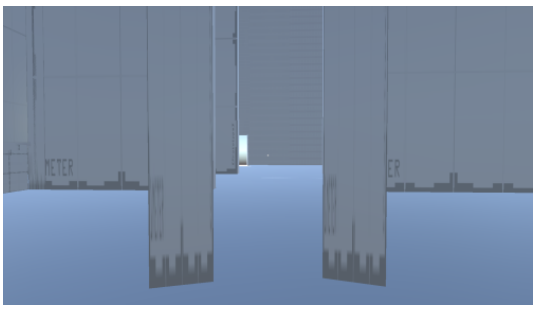

FIGURE 7 – level 2 image 3

FIGURE 8 – Menu Principale

FIGURE 9 – Menu des Options