

2ème soutenance: Dark Angel Reborn

Team TEMP-ète

03/05/2018



Contents

1	Le mot du chef	3
2	rappel du projet	3
3	Realisation de la periode	4
3.1	partie solo	4
3.2	partie multi	5
3.3	partie artistique	6
3.4	Autre	11
3.5	tableau récapitulatif	13
4	cahier des charges pour le prochaine soutenance	13
4.1	partie solo	13
4.2	partie multi	13
4.3	partie artistique	14
5	impressions personnelles	14
5.1	Thomas alias Oberst	14
5.2	Pierre alias Supermanbart	14
5.3	Matthias alias Boulbol	14
6	sources	15

1 Le mot du chef

Certains voient un verre à moitié vide, moi j'en vois un à moitié plein. L'avancement globale du projet est peut-être moins important que celui estimé, mais il reste cependant positif. La partie solo a pris forme et les personnages vivent. Ils ont encore un rôle décoratif mais ils sont capables de se déplacer de manière cohérente et signalent de manière convenable lequel est concerné par un ordre. Ils sont donc prêts pour la dernière ligne droite de l'implémentation. Dans les situations de combat, l'ennemie est clairement déterminée. Il possède un attribut de vie rouge alors que les alliés sont au vert. Il est possible en plus de leur donner des ordres de la même manière qu'aux civiles. Les ennemis suivent un paterne de conquérant, éliminer tous les adversaires est un bon moyen de s'emparer de la colonie. Pour ce qui est de l'ouverture sur le monde, la partie multijoueur a relativement bien avancé. Deux joueurs peuvent maintenant se connecter et se déplacer avec des animations sur la carte. Les combats ne leurs sont pas encore parvenus à cause d'un problème d'identification. Enfin, le monde a découvert la musique, le son en général. Bien que certains détails soient encore à régler, cette partie a bien avancé. Le terrain de jeu s'est vu munir d'un barrage et d'un jet d'eau à l'apparence de cascade. Cela a permis de terminer cet dernire. Le projet est dans la bonne direction. Qu'il ne baisse pas sa vitesse, et il sera prêt pour l'échéance finale.

2 rappel du projet

Ce projet concerne le développement d'un jeu vidéo. Ce dernier peut se définir comme un RTS gestion. Il se décompose en deux parties séparées, sur la même carte, les phases suivantes s'alternent, la gestion de la colonie, et de temps en temps, cette dernière est attaquée et certaines personnes sont capables de la défendre. Les autres sont protégées et ne peuvent être éliminées. Le jeu présente certains aspects techniques. Tout d'abord la réalisation d'un mode solo, présenté ci-dessus, puis la réutilisation de la période de combat dans une version multijoueur en 1 versus 1.

3 Realisation de la période

3.1 partie solo

Les combats La partie combat connaît deux avancées : une partie tir et une partie gestion de la vie des unités. Dans cette partie tir, le premier point est l'ajout d'une distance de tir pour que les ennemis ne puisse pas tirer sur nos alliés dès qu'ils apparaîtront dans le jeu, après cela pour que les ennemis puissent tirer ils faut qu'il se rapprochent du joueur, j'ai donc rajouté un script de déplacement qui va faire déplacer l'ennemi vers l'allié. Le point le plus compliqué est celui de générer un projectile. J'ai donc créé un prefab de ce projectile qui est une sphère de Unity avec un script attaché. Ainsi on va générer ce prefab à une certaine cadence lorsqu'une Unité va être assez proche de ses ennemis. Pour avoir ce projectile je vais utiliser la fonction ‘Instantiate’ qui prend comme arguments : le nom de notre Objet, sa position et sa rotation. Pour sa position, je l'ai localisé sur l'unité qui va tirer mais le souci étant que la balle apparaissait au niveau des pieds de l'unité, cela ne convenait pas. Ainsi j'ai réhaussé sa position y pour qu'elle soit créée au niveau de l'épaule. Le script du projectile commence donc par localiser le joueur le plus proche. Pour cela je crée un tag qu'on va attacher à tout les membre d'une équipe et ainsi ce projectile va pouvoir différencier les différentes équipes. Ensuite on va localiser la position du membre le plus proche de l'équipe et ainsi le projectile va foncer jusqu'à cette direction. Comme pour l'apparition du projectile si nous utilisons la position du joueur, cela envoie le projectile sur le pied de l'ennemi nous augmentons de la même façon la valeur y du point d'arrivé. Le dernier point pour le tir est qu'une fois le projectile arrivé à destination, il faut le détruire pour ne pas surcharger le nombre d'unité dans le jeu. Ainsi on détecte la collision entre l'unité et notre projectile pour pouvoir supprimer ce dernier. Pour détecter la collision on test à chaque frame si la position de notre balle correspond à celle de l'unité visée. Si l'on test ses trois coordonnées (x, y, z) cela crée un souci puisque nous envoyons la balle à hauteur de l'épaule, donc la coordonné y sera fausse et notre projectile ne sera jamais supprimé. Je me contente donc des coordonnées x et z.

La deuxième partie constitue à gérer la vie de nos unités avec une barre de vie. Pour cela on va créer différentes variables : tout d'abord des dégâts associés à notre projectile puis la vie ('Health'). Il y a un problème c'est que je dois associer différents scripts entre eux. Ainsi lorsque notre projectile va arriver à destination de sa cible, l'unité de notre joueur doit retrouver les dégâts causés par le projectile puis perdre des points de vie. Pour cela il faut initialiser une nouvelle variable avec comme type le nom de notre script puis associer ce dernier script dans l'éditeur de Unity. Pour la barre de vie j'utilise l'UI de Unity et je crée un fond noir peu opaque pour voir la vie maximum de notre personnage après avoir perdu des points de vie. Par-dessus j'utilise une autre barre de vie qui va se vider lorsque une unité est touché. Pour cela j'utilise une image de type « filled » et je peux gérer son remplissage dans un script à l'aide de la méthode 'fillAmount' qui nous permet de gérer à quel point est rempli notre barre de vie est remplie. Enfin, la vie n'est pas la seule variable, nous avons aussi la variable 'Maxhealth' qui permet de comparer les points de vie actuel à ceux de départ. Par ailleurs si l'on fait le quotient de Health et de MaxHealth, nous avons un rapport entre 0 et 1 qui correspond à la valeur de 'fillAmount'.

```

5  public class Projectile : MonoBehaviour {
6
7      public float speed;
8      public float damage = 40f;
9
10     private Transform player;
11     private Vector3 target;
12
13     void Start(){
14
15         player = GameObject.FindGameObjectWithTag ("Player").transform;
16
17         target = new Vector3 (player.position.x, player.position.y, player.position.z);
18
19     }
20
21     void Update(){
22
23         if (player.gameObject.activeSelf) {
24             target = new Vector3(player.position.x, player.position.y + 5f, player.position.z);
25
26             transform.position = Vector3.MoveTowards (transform.position, target, speed * Time.deltaTime);
27
28             if (transform.position.x == target.x && transform.position.z == target.z) {
29
30                 DestroyProjectile ();
31
32             }
33
34         }
35
36         void DestroyProjectile(){
37
38             Destroy (gameObject);
39
40         }
41
42         void OnCollisionEnter (Collision col){
43             if(col.gameObject.name == "Player")
44             {
45                 DestroyProjectile();
46             }
47         }

```

Figure 1: Script du projectile

3.2 partie multi

Lors de cette deuxième partie du projet, j'avais deux objectifs principaux : faire en sorte que le multijoueur fonctionne au-delà du local, c'est-à-dire pouvoir créer des rooms (maximum 2 joueurs) sur Internet et pouvoir en rejoindre. Le deuxième objectif était d'ajouter des caméras fonctionnelles pour chaque joueur, que l'on puisse contrôler de la même façon qu'en solo mais qu'elles soient indépendantes les unes des autres. Mais aussi d'implémenter les animations en partie multijoueur. Optionnellement, j'ai ouvert des comptes Facebook, Twitter, YouTube pour promouvoir le jeu (YouTube : https://www.youtube.com/channel/UCPVurGu_YCYTaLw0YpApHwg, Facebook : <https://www.facebook.com/Dark-Angel-Reborn-188046881840623>, Twitter : https://www.twitter.com/DA_AN_RE).

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais ne fonctionnait qu'en local ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. On à donc lié notre compte unity actuel au service gratuit (limité à 20 joueurs à tout moment donné), une salle de jeu ne permet pas plus de 2 joueur dans cette dernière (même si on peut changer ce nombre). Avec le dashboard on dispose de plusieurs outils en plus comme le nombre d'utilisateurs en cours...

Pour ce qui est des caméras, c'est une solution assez simple : il n'y a pas à instancier une caméra par personnage qui spawn, mais une seul caméra dès le lancement du multi. Effectivement les autres joueurs se fichent de votre caméra, on peut



Figure 2: La page Twitter

donc se contenter de les garder (et de les contrôler) sur la "scène". Je n'y avais pas pensé au début et j'avais fait comme pour les déplacement des personnages, la caméra était instanciée avec le personnage et était source de problème.

D'autre part les animations marchent bien en solo mais ce c'était pas le cas dans le multijoueur lors de la première soutenance, effectivement on utilisait un composant "Animator" qui animait les personnages lors des déplacement. Mais en multijoueur le client ne voit les animations que de son personnage. C'est pour cela que l'on a mit un "NetworkAnimator" qui comme son nom l'indique fait la même chose mais communique aux autres joueurs quand leurs clients doivent animer les divers personnages.:

3.3 partie artistique

Le barrage L'un des points point pour cette soutenance était la réalisation d'un barrage. Ce dernier a été réalisé avec blender. La raison de l'utilisation de ce logiciel était que le modèle d'un barrage est introuvable sur internet. L'une des réticences à utiliser ce logiciel était que certains modèles venant de blender avaient eu des incidents à l'exportation du fait de leur haute précision et géométrie. Leur rendu présentait des trous dans des toiles tendues par exemple. Unity propose aussi la possibilité d'intégrer directement des modèles au format blender. Mais cette option n'était pas fonctionnelle sur notre moteur graphique probablement à cause d'une incompatibilité des versions entre unity et blender. La réalisation du modèle sur ce logiciel a été beaucoup plus rapide qu'imaginé. La première raison se situe dans le fait qu'un tutoriel de bonne qualité a été trouvé et a expliqué du début à la fin tous les raccourcis et a apporté de nombreux conseils pour une réalisation plus rapide. L'autre raison est que le modèle produit est à base de forme géométrique simple, principalement des carrés et des rectangles. Le barrage a été plus taillé dans une brique rectangulaire et remodelée avec des extrusions importantes. Ce



Figure 3: La page Facebook

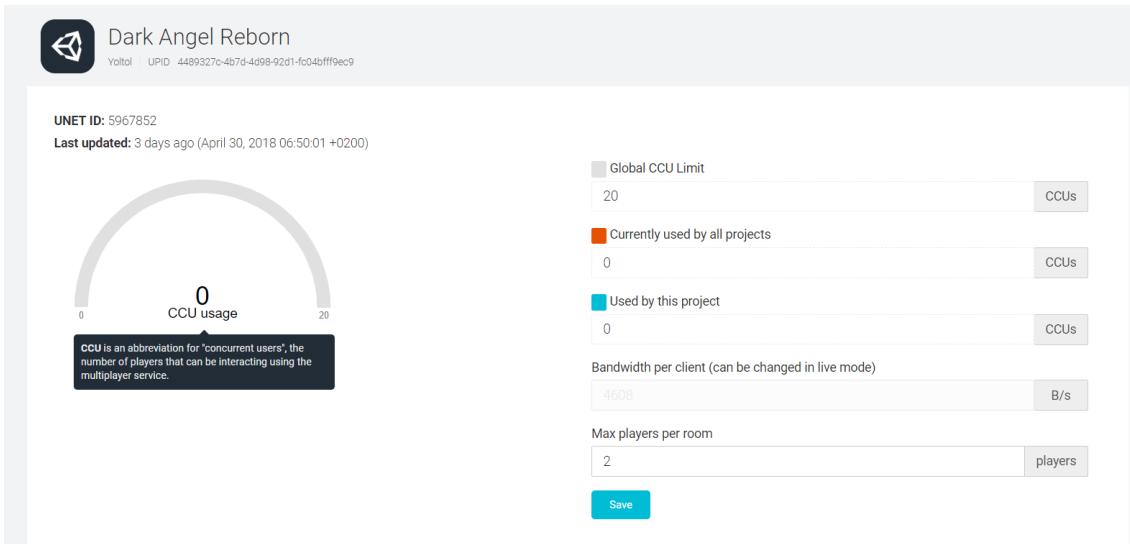


Figure 4: La page Twitter

modèle présente aussi une texture simple, un gris uni, signe de l'état de conservation du barrage, malgré de probable intempéries. Pour simplifier la conception de ce bâtiment, la texture utilisée a été posée avec unity lors de l'installation du bâtiment sur la carte dans la montagne. Cependant ce bâtiment ressort du paysage car il est encastré en flanc de montagne. On peut imaginer qu'il y a un lac derrière le barrage mais le joueur ne peut pas vérifier. La caméra en jeu ne le permet pas. de plus, le barrage relâche continuellement de l'eau, sous la forme d'une cascade. La réalisation de cet élément est présentée dans le prochain paragraphe.

La cascade La cascade ou le lâché d'eau du barrage est l'élément qui lui donne vie. Cette dernière se compose en fait de 5 *particle system*. Deux composent le jet d'eau qui tombe des trous du barrage et les 3 autres constituent l'arrivée, l'impact avec la surface de la rivière. Je présenterai d'abord les caractéristiques des 2 chutes puis les différents points des éléments au sol.

La chute d'eau se compose de deux jets. Le premier représente un flux d'eau

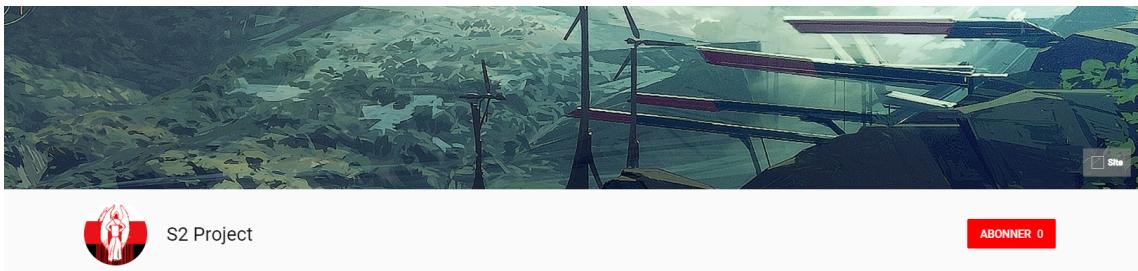


Figure 5: La chaîne YouTube



Figure 6: Le barrage et la chute d'eau

constant, le corps de la chute. La majorité du débit passe par ce chemin. Afin de réaliser ce système de particule, il a fallu réaliser une image. Cette image est un extrait de la représentation d'une chute d'eau de laquelle on n'a gardé que le mouvement de l'eau, les crêtes et les creux de l'eau. Ensuite, l'image a été bordée de noir avec des « langues » de noir avec un large pinceau afin de produire un dégradé important. Cette partie noire lors de l'intégration sera retirée, ce sera comme un fond vert. Le résultat sera une image de forme aléatoire. Cette forme sera utilisée comme particule de base pour la cascade. La vitesse de chute, l'angle et la réaction avec un support seront configurés afin de représenter l'impact de la chute. Une cascade ne se compose cependant pas que d'un flux d'eau. Elle a aussi un brouillard, dans la réalité cet élément provient des particules d'eau qui quittent le flux principal et qui descend plus lentement. Une autre texture a été utilisée dans notre cas. L'image originelle était un brouillard qui a été rogné et traité de la même manière que pour la texture de la chute d'eau. La vitesse de chute des particules est plus lente, et la taille des particules est plus petite. Ensuite, si la cascade tombait sans provoquer de réaction sur son support, l'effet ne serait pas réaliste. Les 3 derniers générateurs de particules sont prévus à cet effet. Le premier va simuler les ondes de l'eau quand les deux flux rentrent en contact. Cela va être un effet d'échappement depuis un point central vers l'extérieur. Les formes de ces ondes sont de demi-cercles orientés vers l'extérieur. Ces particules se propagent de manière parallèle à leur support. Les

deux derniers effets illustrent un nuage de particule d'eau en suspension à l'arrivé de la cascade. C'est l'eau qui saute à cause du flux entrant. Ces deux systèmes de particule reprennent les 2 textures du flux principale. L'un constitue la brume, l'autre l'eau. Ils sont dirigés vers le haut et sont beaucoup plus faibles que ceux de la chute principale.

Tous ces effets ont pour objectif de produire une chute d'eau réaliste. Le dernier point cette réalisation concerne le son de la chute, l'impact de l'eau à l'arrivé.

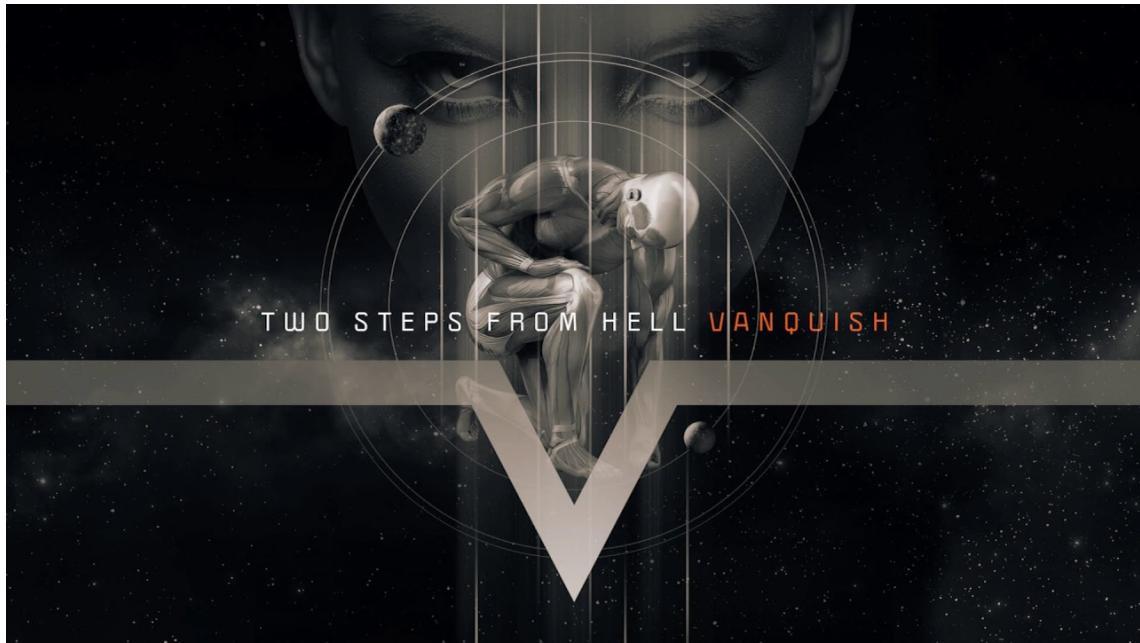


Figure 7: La couverture de l'album Vanquish

Les musiques Il y a dans le jeu deux types de musique. Tout d'abord la musique d'ambiance, une mélodie simple qui cherche à immerger le joueur dans la scène. Et il y a ensuite la musique environnement, par exemple le bruit d'une cascade. Ces deux musiques ne se définissent pas de la même manière, la première est une musique omniprésente dans la scène, qui ne varie pas d'intensité. Le deuxième type de musique est composé de son d'ambiance. Chaque son d'ambiance a une source et plus le joueur va s'en éloigner, plus le son sera faible. Dans unity, le moyen le plus simple de placer un son est d'utiliser un *Component* qui s'accroche à un objet et de le faire joueur. Ce component peut produire un son 2D (omniprésent) ou 3D (qui varie avec la distance). Il suffit donc de placer les bons éléments au bon endroit.

Les musiques utilisées dans ce projet ont été retouchées, afin de mieux correspondre à la situation. Certaines de ces dernières commençaient trop tard par rapport à un cinématique ou étaient trop fortes. Pour les sons d'ambiance, ils sont tous de même provenance, le site www.universal-soundbank.com/ qui propose gratuitement un large choix de son d'ambiance. Tous les éléments, sons ou musique seront listés à la fin de ce dossier.

Il est cependant important de relever que le jeu n'est à l'heure actuelle pas très fourni en son d'ambiance car unity a soulevé certains problèmes et particulièrement dans la spatialisation de certains sons. Cela a provoqué une cacophonie dans le jeu, le son d'ambiance de la mare étant mélangé avec celui de la forêt et le tout à un

volume trop élevé. Ces sons d'ambiance ont disparu que lorsque les fichiers sources ont été supprimés. Toutes les autres options n'ayant pas fonctionnées.

Les cinématiques Le jeu propose au joueur deux cinématiques. La première sert d'introduction et a pour vocation de présenter le terrain de jeu du joueur. Elle se lance au début du jeu de manière automatique et se compose uniquement d'une caméra qui va survoler l'aire de jeu. Cette cinématique est en fait simplement une animation qui se joue et qui a été posée sur une caméra. Lors de cette animation il est possible d'accrocher des événements, au cours de l'avancement de la cinématique. Dans ce cas, il fallait désactiver les différents éléments de l'interface. En effet, une cinématique se joue en pleine écran, sans que la minimap ne soit active ou que les différents éléments de « contrôle » soient visibles. Il faut cependant que les joueurs y aient accès après la première cinématique. La carte étant relativement grande, ils se perdraient rapidement. L'événement qui est donc lancé pendant l'animation est l'activation des éléments de l'interface à la fin de cette dernière. La cinématique se lance au début du jeu, la caméra principale et l'interface sont désactivés à l'entrée de la scène.



Figure 8: Les soldats ennemis sortant de couvert

La deuxième cinématique ouvre le mode de combat du jeu. Il se décompose en deux parties, d'abord une première caméra filme les ennemis qui apparaissent derrière un bâtiment, puis une seconde montre les alliés sortant du bunker contenant les armes. Ces deux caméras ont été relativement plus compliquées à coder car elles demandent des événements plus complexes que la première. Lors de la cinématique, (qui se lance lors que l'on presse une touche du clavier pour cette présentation) des soldats doivent apparaître et se déplacer puis disparaître afin de ne pas surcharger la carte. Ces personnages ont été formés à partir des personnages déjà existant et se sont vu attribuer un nouveau script. Ce dernier active leurs animations et leur assigne une destination. Le *NavMesh Agent* s'occupe ensuite de les déplacer. Pour le groupe des soldats ennemis, ils avaient le temps de la deuxième cinématique pour disparaître, alors que les alliés eux doivent disparaître en même temps que

leur caméra. Au départ, le script attendait que les personnages soient suffisamment proches de la destination pour les désactiver mais il s'est avéré plus simple de les désactiver avec la caméra, en transformant le script de disparition en lui demandant de faire disparaître une liste de *GameObject* au lieu d'un seul. Ces deux caméras ont donc chacune deux événements, l'un qui fait disparaître l'interface (1ere caméra) ou les soldats (2eme caméra) et qui en fait apparaître, les soldats (les 2 caméras) et l'interface à la fin (2eme caméra).



Figure 9: Les colons s'étant équipé ressortent

3.4 Autre

le balisage des personnages Cet aspect présente deux points, le premier concerne la partie civile du projet, l'autre est une exportation de ce modèle avec un ajout pour les combats.

Les personnages sont en général de couleur relativement sombre, dans des teintes de vert avec des paternes algorithmiques afin de produire un camouflage dans des zones de combat. Cela corrobore l'histoire initiale, mais il est plus difficile de les repérer sur la carte. Au départ, les joueurs se repéraient grâce à un polygone vert à la manière des sims au-dessus de leur tête. Cependant cette méthode ne permettait pas de les repérer précisément sur la carte, car il fallait déterminer leur position par rapport au polygone. La solution qui a été trouvée, est celle de poser une sorte de projecteur au sol. Le problème était de trouver ce genre d'objet, car les masques sur unity ne sont très faciles à réaliser. C'est dans le système des particules que se trouve la solution. Un layer des particules retire la partie sombre. C'est une sorte de masque alpha. Il a donc suffi de produire une image avec un disque dessus. Et de lui appliquer ce layer. On applique la texture sur un plane et il ne reste plus qu'un cercle vert au sol. Cela permet deux choses. La première est de ne pas cacher un possible personnage derrière le polygone, et ensuite, le sol s'éclairci de manière importante et d'un vert très clair et lumineux. Les personnages ressortent alors éclairés par le sol et sont beaucoup plus visibles.



Figure 10: les balises civiles

Ensuite, ce système de repérage est aussi appliqué en combat. Les personnages, quand ils sont sélectionnés ressortent en vert. Ces derniers sont aussi équipés d'une barre de vie. C'est un *canvas* qui est personnel et qui flotte à la place du diamant vert qui servait à repérer les soldats avant. Cette barre de vie est soit rouge pour l'ennemi, soit vert pour les alliés. Ce choix est volontaire et tactique ; il est en effet plus facile de reconnaître les alliés et les ennemis ainsi. Pour que ces barres de vie soient progressives, un script définit leur taux de remplissage comme le rapport entre la vie maximum et la vie actuelle, le maximum du rapport étant 1. La barre de vie n'est pas actualisée de manière constante, elle se modifie que quand le personnage prend des dommages. Cela permet une légère optimisation. Enfin, les canvas étaient fixes au départ. Cela signifiait que lorsque les personnages tournaient, les barre de vie aussi. Afin de palier ce problème, une ligne de commande a été rajoutée, elle rend la rotation du canvas plus libre et lui impose de toujours regarder vers la caméra, commande *FaceTo*.

l'écran de chargement Lors de la dernière soutenance, il est apparu que l'écran de chargement du menu principale se figeait lors du passage d'une scène à une autre. Les particules se fixaient et il était facile de supposer que le jeu avait crashé. Mais ce n'était pas le cas, le jeu changeait juste de scène et elle était volumineuse donc longue à ouvrir. Ce problème a été résolu, en créant une nouvelle page dans le menu. Cette dernière annonce le niveau de chargement de la scène et reste dynamique. Pour ce faire, lors du passage de l'écran solo à la scène suivante, une nouvelle fenêtre s'ouvre. Le script de cette dernière va réaliser deux actions en simultanée. C'est une *coroutine*; la scène actuelle n'est pas fermée dès que l'instruction de passer à la scène solo est donné, mais une méthode va récupérer l'avancement de la progression du chargement de la scène, et va la renvoyer. Cette valeur est récupérée puis définis le taux de remplissage d'une barre de progression. De plus, un texte explicite l'utilité de la page, « loading » puis le taux de chargement. Tous ces détails ont pour objectif de conforter le joueur dans le fait qu'il va jouer à un jeu et que son PC n'a pas planté.

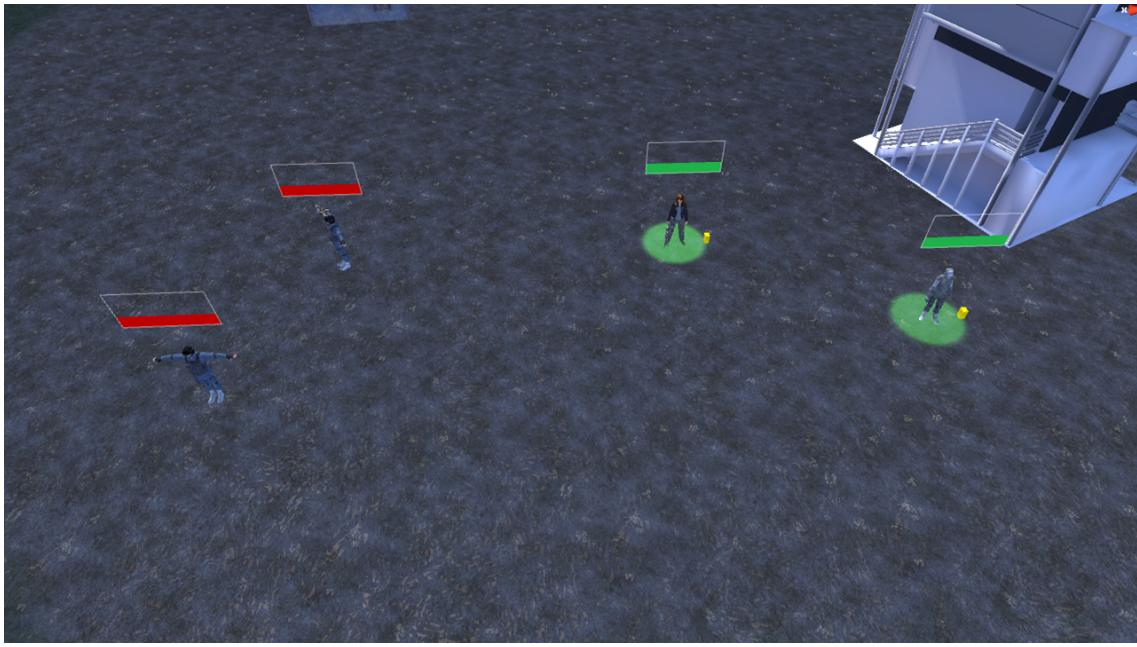


Figure 11: Les barres de vie

Enfin, l'autre aspect de ce script, est que l'ensemble des particules qui traversaient l'écran, ne se figent plus lors du changement de scène. L'image se fige et ne se change que lorsque la scène suivante est déjà chargée donc il n'y a plus de freeze de l'écran.

3.5 tableau récapitulatif

Date	prévu	réalisé
Art	90 %	85 %
Life	50 %	50 %
Attack	60 %	70 %
Multi	80 %	50 %
Stress	50 %	Top SECRET

4 cahier des charges pour le prochaine soutenance

4.1 partie solo

Il faudra généraliser le script fait sur une Unité à plusieurs Unités notamment grâce à un principe de sélection. Le deuxième point à aborder est la gestion de ressource qui ferait de notre jeu un jeu de stratégie et enfin les ennemis ne fonceront plus vers le joueur mais iront dans la bonne direction et donc auront un vrai objectif.

4.2 partie multi

Il faudra implémenter les combats en multijoueur, qui fondamentalement dépendent de trois scripts, mais qu'il faudra les adapter aux besoin (en passant par la méthode transform de unity) pour communiquer aux autres clients les informations nécessaires à la gestion des dégâts, ainsi que les conditions de fin de partie. Je songeais aussi à utiliser l'API de Twitter et/ou celle de Facebook pour partager une victoire ou autre et éventuellement augmenter le nombre de personnages dans le multijoueur qui peuvent se gagner au fil du temps ou autre.

4.3 partie artistique

La majeure partie de cette section a été réalisée. La prochaine période verra la constitution d'un environnement sonore, et la correction de tous les bugs qui pourraient apparaître pendant la période. L'effort de travail va se porter principalement sur le développement de la partie solo et du multijoueur.

5 impressions personnelles

5.1 Thomas alias Oberst

Ce projet est un excellent moyen d'avoir un aperçu du développement d'un jeu vidéo. En équipe restreinte, mais pas en solo quand même. Il faut que les parties concordent et que les modèles soient identiques afin que le résultat soit là. Une seule différence peut faire planter le logiciel. Il est plus facile ensuite de pouvoir communiquer directement soit par téléphone soit face à face directement lors de l'implémentation des nouveautés, cela apporte de la fluidité dans les échanges. Contrairement à la dernière soutenance, pour laquelle j'avais plus produit des éléments artistiques et utilisé des logiciels afin de produire des personnages par exemple, cette période s'est plus définie par écrire des lignes de code. J'ai trouvé cela beaucoup plus intéressant car on n'est pas dépendant d'un autre qui pourrait avoir raté son code, je pense par exemple au logiciel Fuse qui plante à chaque fermeture et qu'il faut réinstaller. Nous sommes en capacité de trouver nos erreurs et de les réparer. De plus il y a de nombreux moyen de parvenir au même résultat. Je suis content de mon équipe et du travail réalisé. Ce projet me fait travailler avec des logiciels que je n'aurai pas utilisé autrement (je pense à blender) et le travail prend forme de manière convenable.

5.2 Pierre alias Supermanbart

Le projet en groupe est vraiment très intéressant comme travail, un travail de groupe sur le semestre est une première pour moi et je trouve vraiment bien d'avoir fait ça avec le groupe car chacun a sa partie prédefinie mais toutes les parties sont liés entre elles, ce qui implique pour chacun d'entre nous une grosse part de responsabilité pour notre partie. La recherche d'information est mon point préféré du projet, c'est à dire rechercher un moyen de faire ce que l'on veut faire. On trouvera toujours des tutoriels sur n'importe quoi mais jamais sur notre cas précis et c'est exactement ça qui est fascinant : adapter et mixer les codes et idées des autres internautes pour pouvoir réaliser notre projet. Réaliser la partie solo est très intéressant, car on se rends compte de comment les jeux vidéos auxquels je joue régulièrement fonctionnent, cela me fait prendre du recul sur tout ce qu'il y a derrière un jeu et tout ce que cela implique. Enfin, je suis satisfait du groupe en général, il y a une très bonne entente dans le groupe et tout le monde travaille afin que le projet arrive à terme.

5.3 Matthias alias Boulbol

Je suis toujours satisfait du groupe, on peut communiquer sans soucis ce qui est quand même pratique. Je considère avoir remplis dans la majorité ma partie multijoueur. Créer des rooms, pouvoir les rejoindre, les animations, la communications de l'équipe du jeux sur les réseaux sociaux, les caméras pour les joueurs. Cependant je suis déçu pour le combat car les scripts de combat ont été écrit avec le solo en tête et non le multi, l'implémenter en multi ne marchais pas, on a préférer l'enlever du

multi pour l'instant pour cette raison. Je pensais également qu'on utiliserais plus GitHub que ça, mais disons que la logistique n'est pas notre point fort.

6 sources

musique des cinématiques:

Aggressive War Epic Music Collection! Most Powerful Military soundtracks 2017
[retouché]

musique principale du jeu:

compass de two step from hell