

Soutenance finale: Dark Angel Reborn

Team TEMP-ète

30/05/18



Contents

1	Introduction	3
2	La partie artistique	7
2.1	Le logo	7
2.2	Les cartes	7
2.3	Le barrage	9
2.4	La cascade	10
2.5	Les personnages	11
2.6	Les armes	13
2.7	Les animations	14
2.8	Les cinématiques	15
2.9	l'écran de chargement	16
3	Le gameplay Gestion	17
3.1	Le personnage	17
3.2	La récolte des éléments	18
3.3	Les assignations de poste	21
3.4	Les issues de la gestion	22
4	Le gameplay Combat	24
5	Le réseau	25
5.1	Le multijoueur : avancement	25
5.2	La communication : avancement	26
5.3	Ce que j'ai fait jusque ici	26
5.3.1	Le multijoueur	26
5.3.2	Le site	27
5.3.3	Communication	27
5.3.4	Extra	28
6	Conclusion	29
7	Sources	30
7.1	modèles 3D	30
7.1.1	les bâtiments	30
7.1.2	les armes	30
7.2	Son et musique	31
7.3	les visuels:	31
7.3.1	le logo	31
7.3.2	les polices	31
7.4	les sources d'inspiration	31
8	Annexe	32

1 Introduction

Lors de notre rentrée, il nous a été annoncé que nous devrions produire un projet informatique en équipe. Très rapidement, des idées de sujet ont été avancées et le projet se dirigeant plus vers les jeux vidéo, une histoire a été trouvée, inspirée d'une série. L'équipe s'est ensuite constituée, Thomas en tant que chef de projet, Mathias se focalisant sur la partie réseau et connexion, vient ensuite Elie qui semblait s'approprier le gameplay civil du jeu et enfin Pierre qui se mit en quête de produire un mécanisme de combat. A partir de ces éléments, un cahier des charges a été présenté et accepté sans aller retours entre Krisboul et l'équipe. L'équipe avait alors un nom, TEMP-ète et un projet, Dark Angel Reborn. Le nom d'équipe est un mélange entre nos initiaux (Thomas Elie Mathias et Pierre) et le mot tempête. Le nom du projet est inspiré de la série Dark Angel, produite dans les années 2000. Cette série a été produite par James Cameron et malheureusement s'arrête avant la fin du scénario originel. Afin de présenter le jeu, il est indispensable de parler de la série. Dans un Los Angeles dévasté à cause d'une IEM, un groupe de personne génétiquement modifié à l'apparence normale essaye de survivre, poursuivis par leurs chefs. Le gameplay que nous proposons reprend cette situation et imagine qu'un groupe de ces héros s'installe dans un endroit reculé afin de vivre de manière plus calme et plus en sécurité. Pour cela ils s'exilent hors des états Unis et implantent une colonie en réutilisant d'anciens bâtiments.

La Story Line: Notre jeu présente deux parties, une partie solo et une partie multijoueur. La partie solo se décompose en deux phases, la première est la gestion de la colonie. Le joueur possède un personnage principal, qu'il va déplacer à travers la carte pour récupérer des composants qui permettront de réhabiliter les bâtiments de son entourage. Ces bâtiments peuvent être des maisons, des fermes ou alors la centrale. Ces derniers permettent de recevoir plus de personnage et que ces derniers ne périssent pas de faim. L'objectif du joueur étant de pouvoir accueillir le plus de colon possible. Ensuite la deuxième phase est une défense de la colonie. D'un point de vue historique, Cela s'explique facilement, les ennemis ont eu vent de la colonie spéciale et cherchent à s'en débarrasser en la détruisant. Les colons sont cependant aptes à se défendre et ne se laisseront pas faire. Les ordres des ennemis sont de détruire le QG situé au centre de la colonie. S'ils y arrivent, c'est la fin de la bataille pour les colons. Les ennemis se dirigent par défaut vers le QG. Mais s'ils arrivent à porter de tir d'un colon, ils vont dévier de leur objectif et se rediriger vers ce dernier avec pour objectif de le tuer. La séquence s'initiera quand un deuxième X4 rejoindra la colonie et se terminera quand les ennemis seront tous mort ou le QG détruit. Ces deux phases sont introduites par des cinématiques. Le jeu propose aussi un multijoueur. L'idée au départ était de faire plusieurs colonies et de les mettre en ligne puis de partager comme un jeu de gestion online. Mais, il a ensuite été décidé que le multijoueur sera une reprise de la partie solo de combat mais que cette fois ci, l'ennemi serait contrôlé par un autre joueur qui aurait rejoint la partie.

Un rapide résumé des grands moments de l'année: La réalisation du projet a cependant été changée car l'un des membres de l'équipe a doublé son S1 et nous a donc quitté. Il produira un projet l'année prochaine et ne peut donc pas rester avec nous. Elie a donc quitté le groupe et il nous a fallu réécrire la répartition des tâches. Le plan initial était que 3 membres prendraient la partie gameplay (Gestion, Combat, Multijoueur) et que le dernier commencerait sur la partie artistique du projet,

terminera rapidement sa partie pour la 2eme soutenance (à 90%) et ensuite se retournerai vers les parties des trois autres membres. Cependant avec ce départ, il a été décidé de greffer la partie de Gestion à celle de Combat. La personne en charge avançant alors de son côté avant la deuxième soutenance puis étant rejointe par l'artiste qui aura terminé son travail. Cependant il s'est avéré que combiner les deux parties du gameplay solo était une idée relativement mauvaise car ces deux parties se composent finalement d'énormément de script, ce qui signifie beaucoup de bug potentiel et encore plus de problème. Les deux parties ont somme toute peu avancées pour les deux premières soutenances, Des parties importantes ont été produites, comme les déplacements mais le gros du travail restait encore à faire. Pour la dernière soutenance, la partie art a été presque entièrement mise de côté et le travail s'est exclusivement porté sur le gameplay de la Gestion. Malgré les embûches posées sur le chemin, le projet est prêt pour la dernière épreuve et nous sommes fières de vous présenter le jeu Dark Angel Reborn par l'équipe TEMP-ète.

Les membres de l'équipe: Cette équipe se compose à la date de la soutenance finale de 3 membres, Thomas Blavet en tant que chef de projet et en charge de la partie artistique, vient ensuite Pierre Bergelin qui s'est chargé de la partie gameplay solo et enfin Mathias Deroche qui s'est focalisé sur le réseau et la communication. Ces trois membres vont se présenter par eux même.

Thomas Blavet ou Oberstamfuher Beaucoup de personnes me demandent l'origine de mon pseudo. Il est vrai que si on cherche scrupuleusement ce dernier sur internet, il n'y a rien. Cependant le correcteur orthographique vous réoriente vers un grade militaire de l'armée allemande. L'origine en est simple, il provient d'une bande dessinée, de la seconde guerre mondiale. Le commandant d'un char avait ce grade, Oberstumfuehrer. Le nom ne m'avait pas complètement et donc j'ai retiré une lettre et changé une autre pour finir en un nom avec une consonance plus française. Il est devenu mon pseudo de jeu et particulièrement sur le célèbre World Of Tank¹. J'ai passé un certain temps dessus (mais le rythme a baissé depuis le début de l'année) mais c'est un jeu très vaste en constante évolution. J'ai aussi passé quelque temps sur les jeux Sim City IV² et Ages Of Empire III³. Ces deux derniers m'ont intéressé car ils amenaient à développer une population. Je suis aussi un grand visionnaire de série de science-fiction. A la rentrée j'avais partiellement terminé la série qui m'a inspiré pour ce jeu vidéo. (J'aurai aimé pouvoir plus me rapprocher de cette dernière mais cela s'est révélé compliqué) Mais le cadre de la survie reste présent même si les civils sont plus retirés. Dans cette production, j'ai été le chef de projet, et aussi le chef de la partie artistique.

Pierre Bergelin ou Supermanbart Je m'appelle Pierre. J'ai très peu programmé avant Epita. J'ai commencé à coder en CSS et html afin de faire un site web en 1ère mais j'ai aussi programmé un peu en TI (calculatrice de terminale) avec lequel j'ai programmé un morpion. Pour le projet je me suis occupé du Game Design et plus précisément la partie associée au combat.

Mathias Deroche ou Boubol Ma principale expérience dans la programmation de jeu vidéo vient de ma terminale avec le projet d'ISN. Je programme aussi un peu

¹Jeu sorti en 2009 et produit par le studio Wargaming. Il a regroupé quelque 190.000 joueurs en même temps

²jeu produit en 2006 par le studio Maxis

³Jeu produit en 2005 par les studios Ensemble Game

pendant mon temps libre. Mais rien de conséquent dans ce domaine et le projet m'a permis d'augmenter mon répertoire personnel.

Grâce à cette équipe de choc, le projet a été mené à bien, et la réalisation de ce dernier nous a permis de comprendre certains points importants, pouvant être décrits comme des point de Savoir Faire.

Les objectifs: D'un point de vue pédagogique, on peut relever certains points évidents. Le premier est bien sûr le travail en équipe. Réaliser un projet de cette envergure est difficilement réalisable seul. Du moins pas avec notre expérience du début de l'année. Même après avoir réalisé ce projet, les possibilités sont tellement vastes que cela semble peu réalisable maintenant. Le travail sera bien sur plus rapide. Mais Unity est un moteur de jeu extrêmement complexe et il faudrait avoir du temps pour le comprendre intégralement.

Le premier objectif est donc le travail en équipe. Il y a deux points de vue sur ce sujet. Celui du chef de projet, en charge des dates, de l'avancement et en général du fil rouge des présentations et ensuite le point de vue des autres membres de l'équipe qui eux doivent s'assurer du bon travail du chef de projet et de leur avancement selon le calendrier. Le travail du chef de projet est relativement simple d'un côté, il donne des dates et s'assure que tout fonctionne. Y compris sa partie. D'un autre côté il doit s'assurer que le projet avance et que les dates butoir sont respectées. Ce travail a été compliqué, d'un part du fait de notre organisation et d'autre part car il est parfois difficile de pousser au travail quand on a soi même des échéances plus importantes. D'un autre côté, il y a l'équipe. Ces derniers doivent principalement avancer dans leur projet et jeter de temps en temps un coup d'œil au travail et aux dates fournies par le chef de projet. Il est préférable pour la bonne humeur du chef de projet de l'avertir en avance en cas de retard sur le travail à fournir pour la date butoir et posséder aussi une réserve assez importante d'excuse dans le cas où le projet passerait au second plan de notre attention.

Ensuite, un autre objectif est sans conteste la prise en main de notre outil, Unity. Ce dernier est un moteur graphique très apprécié qui a été à l'origine de jeux comme Pokémons Go pour ne citer que lui. Cet éditeur comprend deux langages de programmation, le Java et le C#. Ce dernier étant le langage étudié en TP à l'école, c'est celui utilisé pour notre projet et l'ensemble de nos scripts sont écrits en cette langue. D'un point de vue strictement du code, il est possible d'affirmer que la notion de classe et d'héritage peut être maintenant estimée comme assimilée. C'est en effet le cas pour la partie gestion qui l'utilise à foison. L'utilisation d'Unity a aussi permis de découvrir, du moins pour la partie artistique, la manipulation d'objets 3D et les animations ainsi que le jeu des textures qui peuvent rendre un jeu beau ou moche. Il a aussi été possible d'entrevoir la complexité du montage d'un réseau internet pour un jeu avec les différentes problématiques de gestion des clients et du serveur.

Le plan: Ainsi présenté le jeu, les membres de l'équipe et enfin les objectifs supposés de ce projet, il ne reste plus qu'à présenter la répartition de la présentation du travail à travers le dossier et de vous souhaiter une très bonne lecture. Pour commencer, la partie artistique ouvre le projet. C'est la première partie à être terminée. Bien qu'elle ait été terminée en premier, elle occupe une place importante dans le projet car elle en fait sa beauté et son immersion. Vient ensuite la première partie du gameplay, la Gestion. Cette partie vient ensuite car c'est la première chose que fera le joueur quand il arrivera sur le jeu. Ce dernier quand il aura suffisamment avancé

se base enclenchera le gameplay du Combat qui vous sera alors présenté. Et pour finir, le dernier chapitre recouvre la connexion avec l'extérieur du projet. Cela inclus le gameplay multijoueur et la communication faite autours du jeu.

2 La partie artistique

Dans cette partie, l'ensemble des éléments décoratifs et la production des différents effets vous seront présentés. Ces derniers sont principalement orientés vers la beauté visuelle du jeu et l'immersion. Ils influent peu sur le gameplay mais sont les premiers éléments que le joueur remarque (ou pas).

2.1 Le logo

La réalisation du logo a été la première partie à être avancée. Il représente l'ouverture du jeu et se rapproche plus du titre que du gameplay. Ce logo a deux principales sources d'inspiration : la première est le drapeau des augmentés dans la série Dark Angel. Il avait été créé en signe de paix et représentait la progression vers la paix, objectif de ces gens. La seconde source d'inspiration vient du titre, dark ANGEL reborn ; la partie de l'ange m'a fait penser à retirer la colombe symbole de la paix par une faucheuse, se rapprochant plus du jeu de combat. Il avait été question à un moment de placer dans les mains de cet ange de la mort un fusil d'assaut. Mais il s'est avéré à la réalisation que l'ange les mains dans le dos était plus intéressant. Tout d'abord car l'intégration de l'arme avec les manches de ce dernier aurait probablement réduit la clarté du dessin. Ensuite, il y avait un problème de couleur, le fond étant composé de 3 couleurs dominantes, noir, rouge et blanc, afin de bien se détacher, l'arme aurai du-t-être en bleu ou en orange. Des couleurs peu intéressantes pour une arme en soit.

L'ange a donc perdu son G36. Mais il a gagné une posture d'attente, de surveillance, presque une attitude paternelle. Le rôle de la colonie peut y être assimilé, surveiller les créatures de Manticorp et les accueillir si elles en ont besoin. Dans la série originelle, les héros utilisaient rarement des armes. Ils n'en avaient soit pas besoin, car ils étaient des armes vivantes. Cet aspect n'est pas totalement pris en compte dans le jeu mais cela serait un excellent point d'amélioration après le projet. L'ange a cependant un faux derrière lui. C'est son signe de la mort. Les X ou les W sont des soldats avant tout et donc ont une capacité à semer la mort non négligeable. Le logo rappelle donc les X, mais aussi leur objectif d'attente et de fraternisation.

2.2 Les cartes

La réalisation des cartes a été un des points importants de cette période. Le jeu propose deux cartes. La première pour la campagne solo. La deuxième est réservée au mode multijoueur. Sur ces cartes, on peut retrouver des bâtiments similaires. L'objectif de la carte multijoueur est de pouvoir fournir une carte plus petite pour permettre un téléchargement plus facile. Cependant les principales structures restent présentes. Leurs agencements sont juste différent. Certaines structures ont été téléchargées sur un site de modèle 3D. Au départ, je souhaitais utiliser des modèles blender mais les rendus n'ont pas fonctionné. Il y avait un bug sur le rendu lors de l'exportation et donc des surfaces normalement planes devaient striées. L'utilisation des fichiers blender directement n'a pas fonctionné non plus car le blender utilisé devait être trop récent par rapport à la version de unity utilisée. Donc certains bâtiments ont été pris du site <https://free3d.com>. Les références des personnes ont été gardées et sont données à la fin du document en source.

La carte de la campagne solo est grande (1000*500). Cela produit un dossier très lourd. Sachant qu'il y a près de 5.000 arbres plantés sur la carte. Cette carte est

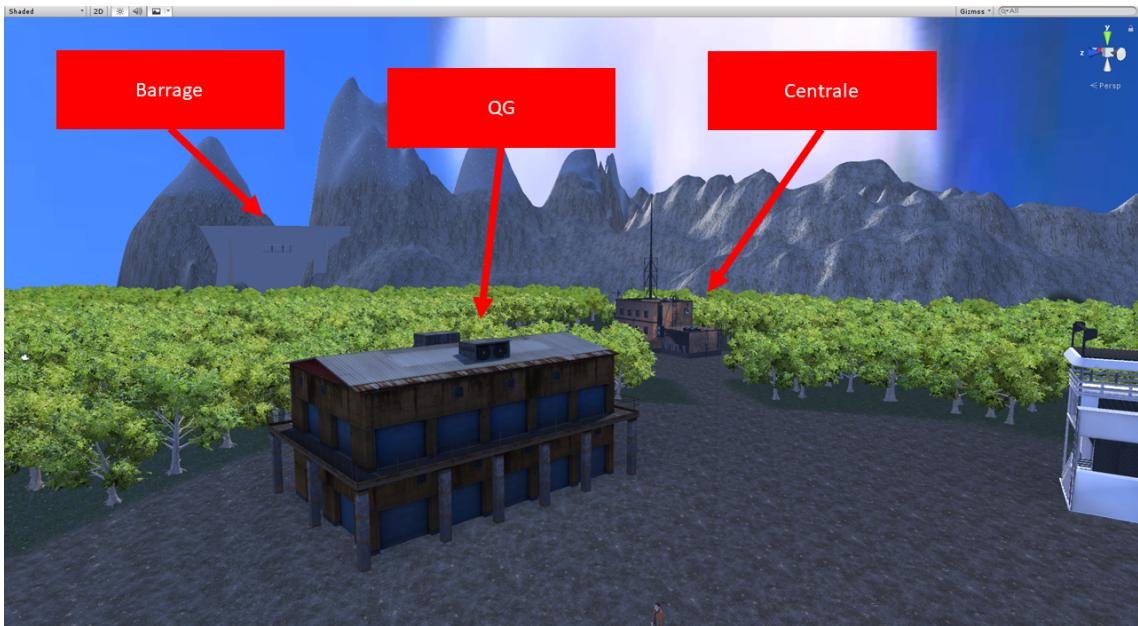


Figure 1: La carte solo

proche de celle décrite dans le cahier des charges, une zone centrale avec l'entrepôt qui sert de QG, il y a autours une maison qui permettra de faire vivre un certain nombre de personne. Il y a aussi une tour blanche. Cette dernière est tirée du site web. Elle sert de mirador pour voir arriver les combats. Il est aussi possible de l'utiliser comme maison (le dossier dans laquelle se trouve ce modèle le présente comme une tour maison). Cette zone est en contact avec 4 zones qui orbitent. Ces zones possèdent des maisons presque toutes différentes, et des bâtiments spéciaux. Dans la liste de ces derniers, il y a une structure équipée d'antennes et de couleur marron métal rouillé. Ce bâtiment sera la centrale énergétique de la colonie. Elle pourra être améliorée pour produire plus d'énergie via le barrage. Il y a ensuite deux tentes de combats. Ces dernières serviront l'infirmerie. Elles entreront en fonction après les phases de combat, lorsque les alliés neutralisés seront rapatriés au camp. Il y a pour finir, deux bâtiments de ferme. Ce sont des bâtiments composés de 3 murs et d'un toit de tôle. Ils permettront la production de nourriture pour la colonie. En plus de ces structures, Il y a un certain nombre de caisses qui sont dispersées autour de la colonie. Elles apporteront des éléments d'amélioration à la colonie comme annoncé dans le cahier des charges. La carte est très boisée cependant, le joueur peut passer entre les arbres. Cette capacité devra probablement être revue dans le cadre de la vie civile, les personnes devant plus utiliser les chemins. Mais cela composera un script pour le life. La carte solo présente aussi un lac, les joueurs ne peuvent se déplacer dans ce dernier. Pour cause, la déclivité des bords est trop importante et donc le jeu lors de la compilation des zones de déplacement l'a retiré.

La carte multijoueur est une carte de taille plus modeste et carrée (500*500). Il avait été pensé un moment de reprendre la carte du mode campagne pour le multijoueur mais au vu du temps de chargement de cette dernière sur un ordinateur non démunis de capacité, il s'est avéré qu'il fallait un nouveau terrain. Cette carte est donc de taille plus modeste. L'entrepôt se trouve au milieu de la carte, entouré de maisons et de 3 bâtiments spécifiques. Ces derniers sont la tour de guet, l'une des tentes d'infirmerie, et enfin le bâtiment pour la production d'énergie. Dans ce mode de jeu, les ressources ne seront pas prises en compte. La disposition de cette

carte est différente de celle de campagne, d'une part à cause de sa taille et d'autre part, car elle a vocation au combat et non à la découverte de zone. Les bâtiments apportent donc des couverts aux soldats, mais restent suffisamment espacés dans l'optique d'un sniper. Il y a certains bosquets dans la colonie, tout d'abord pour l'esthétique, puis cela permettra de cacher les soldats de leurs ennemis en cas de besoin.

Ainsi nous sommes en présence de deux cartes d'un climat plutôt tempéré, avec des forêts denses et vertes. Les deux cartes sont différentes par leur disposition et par leur utilité. La carte de campagne vaste et visitable, la carte de multijoueur ayant pour objectif les combats. Ces cartes pourraient être plus chargées en décors comme des déchets au sol, des feux. Mais leur taille actuelle déconseille fortement d'augmenter le nombre de décors si on souhaite encore jouer de manière fluide sans attendre 45 minutes de chargement. Les bâtiments nous ont permis de diversifier les structures sur la carte, et ont apporté un bon compromis entre rendu visuel et taille du dossier du projet.

2.3 Le barrage



Figure 2: Le barrage

Le barrage est un élément complètement personnel réalisé avec blender. La raison de l'utilisation de ce logiciel était que le modèle d'un barrage est introuvable sur internet. Cependant 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'imagine. La première raison se situe dans le fait qu'un tutoriel de bonne qualité

a été trouvé⁴ 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 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.

2.4 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 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 descendent 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'arrivée 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'arrivée.

⁴La chaîne se nomme TouTApprendre et la série se nomme "comment débuter sur Blender"

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 jouer. 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.

Les sons d'ambiances ont posé certains problèmes lors de leur installation. Mais finalement, leur mixage final a permis de former une ambiance d'une vie dans la nature même si les forêts sont vides d'animaux vivant (humains exceptés). Ces derniers auraient été invisibles à l'œil nu et peu utiles dans le jeu sauf pour consommer plus de puissance d'un ordinateur déjà trop chahuté. Les sons d'ambiance se décompose en deux parties, ceux d'origine humaine, et ceux de la nature. Ceux naturels sont fixés à des endroits qui leur correspondent (afin de ne pas mettre un marais dans une forêt de conifère). Ces derniers sont faciles à lister, les bruits de foret, les bruit de lac et rivière, les bruits de chute d'eau et enfin les bruits de bâtiment. Viennent ensuite les bruits des personnages. Dans cette catégorie prône en maître le coup de feu. Les explosions liées au combat sont très bruyantes. Ce sont même les seuls bruits de combat qui interviennent en fait. Les bruits de pas sont négligeables et les ordres oraux ne sont prévus pour cette version. Pour les coups de feu, il y a un ensemble de bande musicales, des tirs de rafale de différentes armes, selon le calibre les sons sont plus graves, plus espacés. Il est important de noter que ce ne sont pas de bandes sons d'un seul coup mis en rafale mais une seule bande audio pour toute la rafale. Ces éléments ont pour objectifs de signaler au joueur quand ses unités ouvrent le feu et pour le mettre dans l'ambiance des combats.

2.5 Les personnages

Tous les modèles du jeu ayant été créés et intégrés. Il a fallu définir les shaders de ces derniers. En effet le shader par défaut était transparent, nous avions un aperçu des organes internes des personnages et de gros problèmes pour les visages. Afin de différencier les personnages et donc leur fonction, nous pouvons nous repérer aux couleurs de leurs vêtements.

La réalisation des personnages et de leur habillement s'est faite via le logiciel Fuse. Ce logiciel a connu son heure de gloire vers 2015 mais ne semble plus très soutenu par ses créateurs depuis 2 ans. Le logiciel d'abord disponible sur Steam est maintenant exclusivement utilisable sur adobe créative cloud. Pour trouver ce dernier dans ce cloud, il faut passer l'interface du cloud en anglais, Fuse n'étant pas référencé dans la liste française ou alors il n'est pas traduit dans la liste. Après un téléchargement plus ou moins long (selon la localisation et la présence de fibre optique dans le bâtiment) où vous téléchargez le logiciel, Fuse s'ouvre via l'icône sur



Figure 3: Différents type de personnage

le bureau et pas autrement. Il faut ensuite tout faire pour que le logiciel ne se referme pas car il ne peut se rouvrir que si on le retélécharge. Après cette présentation de la méthode pour ouvrir le logiciel, son utilisation après est relativement simple, il suffit d'assembler des parties de corps pour former au choix un homme ou une femme (il est possible de mixer les deux mais cela reste déconseillé). Après le corps physique, le logiciel nous propose un ensemble de vêtements à appliquer sur le corps. Cela va des robes aux gilets tactiques en passant par les casquettes et les cheveux. Donc les personnages ont pris forme avec Fuse, ils ont ensuite été exportés vers le site mixamo qui leur a créé un *rig*. Ensuite, une liste assez importante d'animation nous est proposée par le site. Le choix des animations et leur liste vous sera proposée plus loin, je ne m'avancerai pas plus dans le sujet. Lors de l'importation du personnage dans Unity, il a fallu vérifier que les textures étaient bien appliquées puis s'occuper des animations.

Les héros (les X4) portent un haut de civil, un blouson en cuir en général, et gardent un lien militaire, leur pantalon treillis de couleur vert. Ces derniers sont capables de prendre les armes pour défendre la colonie en cas d'attaque, ils présentent un mixte entre la vie civile qu'ils cherchent à vivre, mais aussi un rappel de la raison de leur création. Ils ont capables de se défendre et du fait de leur modifications génétiques possèderont des capacités supérieures à un soldat normal. Ils peuvent aussi produire des ressources mais en quantité normale (pas particulièrement importantes).

Les soldats (X7) présentent un uniforme complet de couleur sombre, Ces derniers sont incapables de réaliser autre chose que de se battre. Nous avons une femme et un homme. Dans la série originelle, ce ne sont que des enfants qui se battent en groupe de 6 ou 7. Dans le cadre du projet, je n'ai pas dupliqué l'homme à 6 reprises. Pour ce qui est de la femme, elle sert de chef d'escadrons, en effet les X7 devaient être encadrés par des X4 au combat. La femme joue ici le rôle de chef d'unité, elle n'est pas X4 et a donc le même uniforme que les hommes de son type, et ne pourra que combattre.

Viennent ensuite les unités spécialisées dans la production. Ces derniers ont été créées par Manticorp. Ils apparaissent pour la première fois dans la 2ème saison de la série mais forment une partie intégrante de la colonie. Ils sont indispensables pour la production des ressources vitales de la colonie comme la nourriture ou l'énergie. Ils portent un mélange presque identique à celui des X4. Le haut de leurs vêtements rappelle un civil mais leurs pantalons sont des treillis, camouflés. Ce camouflage n'est pas vert foncé comme les héros mais beige, désertique. Leur rôle premier est de produire des ressources, Ils portent donc ces pantalons pour leur résistance lors des travaux. Leurs capacités de combat ne sont pas nulles mais plus faibles que leur « grands frères » les X4. Ces derniers n'ont cependant pas leur capacité de production et il n'y a pas de rivalité entre ces deux groupes du fait de leurs spécialisations différentes mais complémentaires.

Le dernier type de personnage présent sur la colonie sont des civiles. Ils interviennent dans le cadre de mariage. Les X ou W ne sont en effet pas forcés de se marier entre eux et l'amour peut rendre fou (je vous incite fortement à regarder la série, l'amour y occupe une certaine place et permet de nombreuses intrigues aux dénouements plus ou moins heureux). Les amoureux ne pouvant être séparés, certains civils sont invités à titre exceptionnel dans la colonie. Ils sont capables de produire des ressources et peuvent tenir la place d'infirmier. Ils sont cependant incapables de se battre et sont donc obligés de se cacher lors des phases d'assaut de la colonie. D'un point de vue vestimentaire, ils n'affichent pas de paterne particulier.

2.6 Les armes

L'ensemble des modèles ont la même origine que certains des bâtiments des cartes, le site <https://free3d.com>

Ce jeu propose 5 armes. La première est un pistolet, le modèle utilisé est celui d'un Beretta de la police. L'arme présente s'apparente à un Beretta 95, doté d'un chargeur de 15 coups. Le choix s'est fait avec un point de vue artistique de l'arme, de ses formes et du niveau de finition du modèle 3D, comme pour toutes les autres armes du jeu qui vous seront présentées. Cette arme présentera une faible portée, des dégâts modérés, et un poids faible qui rendra le déplacement plus important.

La seconde arme est un FN SCAR-L. C'est un fusil d'assaut Belge. Il a une capacité de chargeur pouvant varier entre 20 et 30 balles selon les versions. Ce détail de réalité quand à la quantité de balle dans le chargeur sera probablement oublié car le rechargeement n'est pas intégré dans les animations. Cette arme peut aussi être dotée d'un lance grenade, d'un silencieux et de nombreux viseurs ou pointeurs. Cette arme possèdera une portée moyenne, de dégât moyen et de poids moyen.

La première arme spécialisée est le fusil de précision M200 CheyTac. C'est une arme de longue portée, doté d'un fort pouvoir d'arrêt. Il est aussi précis et a permis le record de tir groupé à longue distance. Dans le jeu, il ralentira fortement le déplacement du personnage, mais lui permettra de tirer à travers la carte s'il voit l'ennemie. Ses dégâts seront importants. Il possèdera aussi une cadence de tir relativement importante pour un fusil de précision, car il est équipé d'un chargeur.

La seconde arme de spécialisation est une mitrailleuse légère, M60E4. C'est la version la plus fiable de la M60, arme très médiatisée grâce aux films réalisés avec cette dernière. Elle était équipée de bande de munition de 100 à 200 coups. Dans le

jeu, elle sera de portée moyenne, avec des dégâts par coup faible, mais une cadence de tir très important. Elle sera cependant lourde à porter.

La dernière arme de spécialisation est un fusil à pompe, un remington 870. (Numéro supposé car non indiqué par le créateur). Cette arme sera dotée d'une faible portée, d'une puissance de feu très forte, (identique à celle du sniper) et d'un point faible proche de la FN SCAR-L. Si le choix des armes s'est basé principalement sur le designs et rendus visuels, le choix des types des armes était volontaire afin de présenter des styles de jeu différents, il est compréhensible qu'un tireur d'élite devra-t-être placé différemment d'un soldat équipé d'un fusil à pompe. Le poids des armes est pris en compte.

Ci-dessous, un Screenshot des armes, leur style permet de les différencier et si ce n'est pas le cas nous avons de gauche à droite en haut, le M60E4, et à droite le M200. En dessous nous avons le FN SCAR-L, puis le Remington et enfin le Beretta.

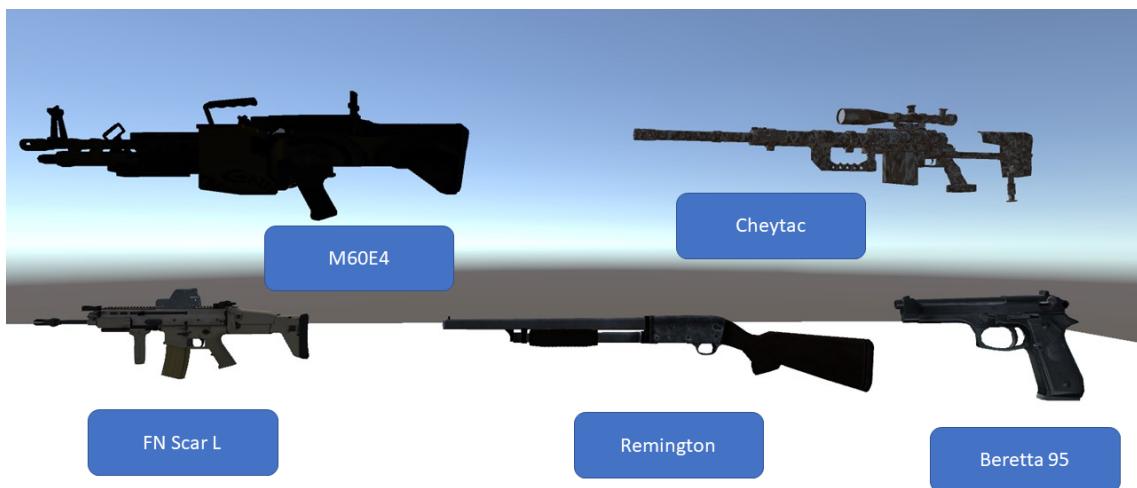


Figure 4: Les armes

2.7 Les animations

Maintenant que tous les personnages ont été présentés, que les armes ont été listées, il faut présenter les animations qui ont liées les armes et les hommes (et femmes). Dans les armes, le pistolet est une arme de point et ne permet pas les mêmes animations que les armes dites longues, toue les autres. Cela impose déjà deux Contrôleurs d'animation. De plus, les civiles n'auront pas d'arme et donc on peut les différencier. Il y a donc dans le projet 3 contrôleurs d'animation ; celui des civils, celui des soldats dotés d'arme longue et enfin celui pour les soldats dotés d'arme de poing.

Le contrôleur des civiles est le plus simple. Le personnage est soit statique, il est en idle. Celui choisi sur le site de mixamo présente un personnage se tenant le dos droit, mais sans bonder le torse, ni prendre une pose trop féminine, le contrôleur étant pour les hommes et les femmes. Ce personnage peut aussi se déplacer, il a donc une animation de marche. Cette animation est lancée avec un booléen, elle est rappelée quand ce booléen est vrai. Et s'arrête quand il est faux. L'animation

idle est donc relancée en boucle tant que le booléen de marche est à faux. Ce booléen est géré par un script qui détecte si le personnage est en déplacement avec *vitesse.magnitude* qui lorsqu'elle est nulle signifie que la personne est à l'arrêt.

Le second contrôleur est celui pour les armes courtes, le pistolet. Ce contrôleur a un idle pour le personnage, le personnage vise. Ensuite il a une connexion avec une animation de tir, le personnage baisse légèrement les bras et tire. Cette animation se lance avec un booléen de tir. Et continue tant qu'il n'est pas remis à faux. Il aurait été préférable de trouver une animation de tir quand le personnage vise mais le site n'en avait pas. Si le décalage apparaît comme trop important, alors l'animation idle pourra être réutilisée et pour signaler le tir, une particule de tir pourra être installé. La personne détenant un pistolet ne se contentera certainement pas de rester statique, le personnage peut donc se déplacer. Le paternité de déplacement est quasiment le même que pour le civil, un booléen de marche et tant qu'il est vrai, l'animation continue de tourner. Le soldat peut aussi mourir. Dans ce cas, un booléen mort est activable, quelque soit l'action, il est prioritaire et lance son animation sans capacité de retour en arrière.

Enfin le dernier contrôleur est celui pour les armes longues. Un soldat avec une arme longue ne peut pas la tenir en joue comme un pistolet; Il y a donc deux animations idle, la première fait tenir l'arme au niveau de la hanche, perpendiculaire à l'axe du torse, c'est l'animation par défaut. Elle est ensuite liée à une animation de visée. Elle est activée par un booléen qui ordonne au soldat de viser. Il y a ensuite deux possibilités, soit le soldat tire, soit il baisse son arme. Le coup de feu est ordonné par un trigger. Ce qui signifie que le nombre de fois que le trigger est vrai, l'animation se déroule. L'animation idle de visée peut être rejouée grâce à un booléen ou elle revient au point de départ. La troisième possibilité est celle de la course, le soldat peut courir avec l'arme pointée devant lui. Cette animation comme celles idle peut être rejouée tant que son booléen est vrai. Dans tous les cas, le soldat peut mourir pendant le processus. Il y a donc à chaque fois un trajet pour aller à l'animation de mort.

Les armes ayant été téléchargées séparément des animations, de petit bug apparaissent, comme lors de la position dite d'attente, arme en travers de la hanche, le soldat tient son arme sur les avant-bras et pas les mains. Ce bug est causé par le fait que l'arme est tenue via un seul point, la main droite. Il est probable que si elle était tenue par deux points, elle serait plus proche de la vérité. Ce choix vient du fait que la position de référence est la position de visée. Position à laquelle l'arme est plus proche de la réalité. Sinon, l'autre possibilité serait de réaliser toutes les animations personnellement.

2.8 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‘ évènement qui est donc lancés pendant l’animation est l’activation des éléments de l’interface à la fin de cette dernière. La cinématique se lançant au début du jeu, la caméra principale et l’interface sont désactivés à l’entré de la scène. Il est possible de sauter cette cinématique à l’aide de la touche échap. Cette possibilité viendrait à avoir lieu si le joueur avait déjà fait une partie et connaissait la cinématique de départ.

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 évènements, 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).

2.9 l’écran de chargement

Lors de la première 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.

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. De plus contrairement à la deuxième soutenance, lors du chargement complet de la barre, la scène s’ouvre directement et il n’y a plus besoin d’attendre.

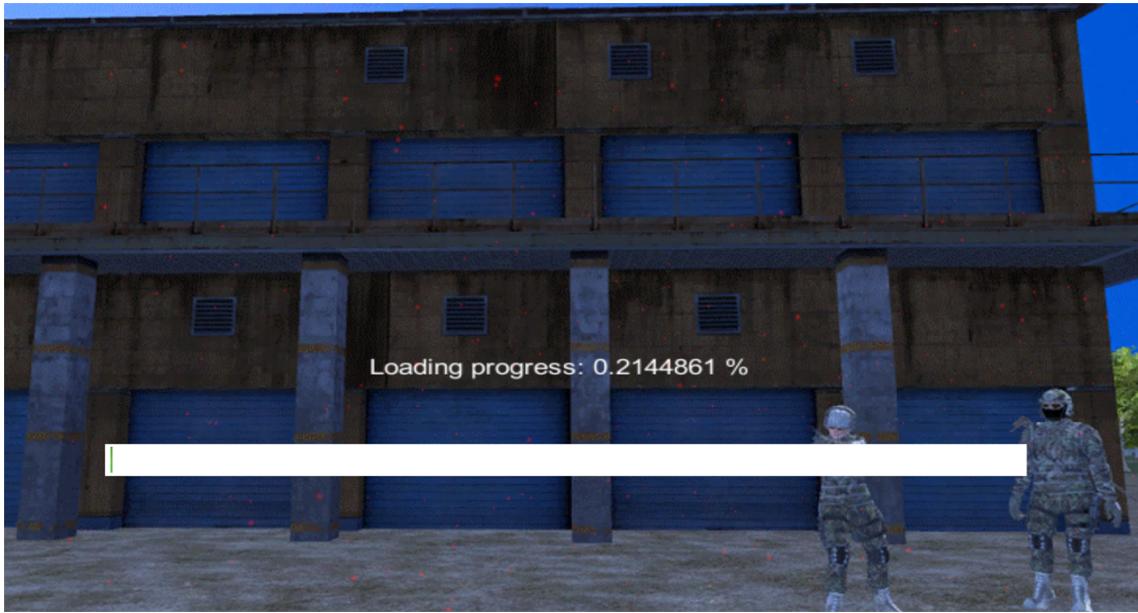


Figure 5: La barre de chargement

3 Le gameplay Gestion

La partie gestion compose la moitié du mode solo. Tout d'abord je vais présenter le personnage puis sa manière de récupérer des éléments. Ensuite, je présenterai l'amélioration des structures. Dans la lancée suivra les assignations des postes de travail et les différentes conséquences de ces actions. Et pour finir il est obligatoire de présenter les différentes issues possibles du jeu dans cette partie. Dans l'ensemble de la partie, je ferai référence à des scripts d'objet en utilisant le nom de l'objet.

3.1 Le personnage

Le personnage a tout d'abord un déplacement. Avec un ensemble d'éléments qui lui permettent de signaler une sélection et une destination. Pour ce faire, il a fallu équiper le modèle d'un certain nombre d'objets. Tout d'abord une balise au sol, un cercle vert translucide à la manière d'un nuage qui sera collé aux pieds du joueur. Ce dernier est formé grâce à un plan qui est posé au sol et qui prend ses positions relativement au joueur. C'est un objet qui est attaché au joueur. Sur cette surface, il a été ajouté une texture d'un point vert avec un fond noir. Puis le type de la texture a été changé pour permettre le retrait du fond et avoir finalement un beau cercle vert à moitié transparent. En plus, une balise de position qui se présente comme un rectangle à la verticale de couleur jaune et rouge. Le cercle sert à signaler que le personnage a été bien sélectionné. Un personnage qui a un cercle vert est le seul à pouvoir recevoir des instructions sur la carte. La balise de déplacement indique la position que vise le joueur. Maintenant que les éléments de déclaration visuel ont été déclarés, il faut expliquer comment le personnage prend ses coordonnées de sa destination. Pour être sectionné, le joueur (vivant) clique sur le personnage avec la souris. Une fonction se lance alors qui analyse si le personnage était déjà sélectionné. Si c'est déjà le cas il ne fait rien de plus sinon il passe à vrai un booléen et active le cercle vert. Le personnage peut alors recevoir des ordres. Le script va lancer une fonction à chaque image qui va analyser le clavier et la sourie

du joueur. Si le joueur utilise son clic droit alors il va récupérer le point d'impact et transmettre la position à un composant du personnage, le NAV Mes Agent qui va tester si la destination est atteignable et qui va calculer alors le plus court chemin. Ce composant va ensuite orienter le personnage de manière à regarder devant lui puis il va changer la position de manière à avoir l'impression que le personnage se déplace de manière humaine. Les animations rendant ce trajet vivant. Quand le personnage aura atteint sa destination, la balise de destination aura la même position que le personnage et alors elle sera désactivée. Si le joueur utilise son clic gauche ailleurs sur la carte, alors le joueur est désélectionné et il termine son chemin puis il attend. Le personnage a aussi la possibilité du fait de son travail d'avoir un ensemble de destination à parcourir comme un paterne. Le perso va donc lire les coordonnées à la suite et s'y diriger avant de passer à la suivante.

Le personnage a aussi un ensemble d'attribut personnel. Ces derniers sont d'une part son employeur et d'autre part sa consommation personnelle. La valeur de l'employeur est une chaîne de caractère. Il aurait été préférable d'utiliser une énumération, mais il n'a pas été possible de faire coïncider ces dernières entre les différents scripts qui vont utiliser ces éléments. Cela implique donc de ne pas se tromper d'un caractère lors de la copie des différents éléments dans les autres scripts mais c'est aussi un genre d'erreur facile à trouver et donc à corriger. L'employeur est au départ une valeur nulle et elle peut être assignée ou récupérée. La valeur de la consommation est propre à chaque type de personnage, les personnages de Manticorps consomment plus que les civils car ils sont différents de leurs congénères. Cette valeur peut seulement être récupérée.

3.2 La récolte des éléments

Maintenant que le personnage se déplace à travers la carte, et est en capacité de récupérer des caisses de ravitaillement qui vont lui fournir des éléments. Ces derniers devraient former une énumération générale partagée avec tous les bâtiments. Mais cela ne s'est pas réalisé, car il aurait fallu commencer par implémenter cette particularité et ensuite déployer les bâtiments à travers la carte. Pour palier cette difficulté, les éléments se présentent sous la forme de chaîne de caractère, des strings et peuvent être transmis entre l'inventaire et les caisses de ravitaillement par un appel de fonction, le processus est simple, le personnage principal s'approche d'une caisse de ravitaillement repérée par une fumé, puis le joueur cloque sur la caisse et un Canvas s'ouvre.

Sur ce dernier est appliqué un texte généré par l'objet ouvert. Cela signifie qu'il n'y a qu'un seul Canvas pour toutes les caisses de la carte et seul le texte change. Ce texte est en fait la présentation du contenu. Une boucle parcourt tous les strings et les accolés en ajoutant un saut de ligne. Puis le texte est assigné à la zone de texte et le tour est joué. Le joueur pour quitter ce dernier doit appuyer sur la touche échap. Une autre possibilité aurait été de créer un bouton de fermeture. Mais la touche échap s'est avérée la plus simple et permet de ne pas surcharger l'image assez remplie. Si le joueur souhaite ouvrir la caisse à nouveau, un autre texte s'ouvre, « vide » car lors de la première ouverture, tous les éléments ont été transférés à l'inventaire. Ce dernier stocke les éléments dans une bibliothèque. La première valeur est le nom de l'élément et le deuxième est un entier qui représente la quantité d'élément qu'il y a dans l'inventaire.

Ces éléments vont servir aux bâtiments principaux comme les fermes pour les améliorer et augmenter leur production. Pour ce faire, le joueur va exécuter à peu

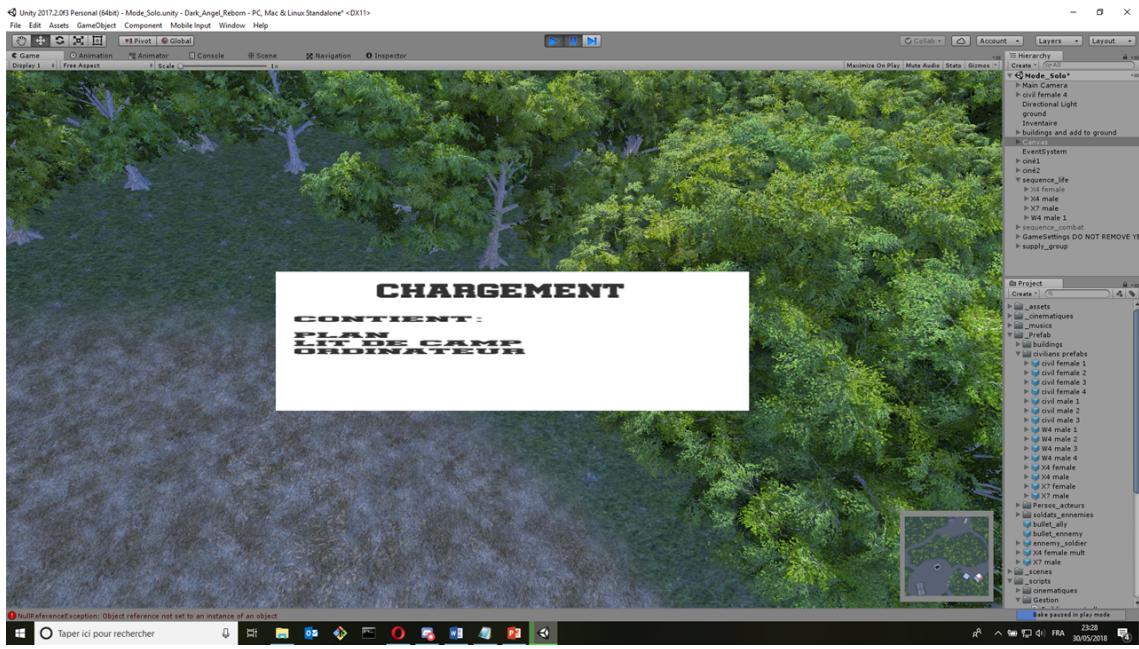


Figure 6: L’interface des caisses

près la même manœuvre que pour les caisses de ravitaillement mais cette fois ci il n’a pas besoin de se rapprocher du bâtiment. Il suffit donc de cliquer sur la structure et l’interface de cette dernière s’affiche à l’écran. Au premier coup d’œil, le joueur est en capacité de comprendre son utilisation. Il y a deux zones de texte, un titre qui est en réalité le nom de l’objet dans le jeu et un bouton d’amélioration. La zone de texte de gauche sert à décrire les caractéristiques du bâtiment, et enfin le niveau de la structure. Il y a le type de ressource produite, puis la quantité de cette dernière. La zone à gauche présente les éléments requis pour l’amélioration. Dans le cas où tous ces derniers sont réunis dans l’inventaire alors ce dernier peut s’améliorer. Dans le cas contraire, le jeu ne réagit pas. Il aurait été intéressant de pouvoir lancer un texte qui signale que l’action a bien été prise en compte mais qu’elle ne peut être réalisée.

D’un point de vue plus technique, il y a deux scripts principaux dans cette section. Le premier est le *contrôleur centrale*, appelé sous le nom de *Inventory_controller* et ensuite, le *Building_controller*. Le building controller est attaché aux bâtiments et l’inventory controller lui à un Empty GameObject « Inventaire ». Lorsque le joueur clique sur le bâtiment, une fonction se lance, ouvre le Canvas et change l’état de la structure avec le booléen *can_be_closed*.

Cette variable au départ destiné à l’interface s’est ensuite avérée très importante car lorsqu’elle est active, elle permet de signaler que les fonctions d’amélioration du bâtiment sont fonctionnelles. L’un des exemples les plus concrets se situe au niveau du boutons d’amélioration. Le bouton est unique pour toute les structures du jeu. Pour les différencier il a alors fallu trouver une variable propre au bâtiment utilisé qui soit différents des autres. Le booléen *_be_close* est le seul à être actif dans le script quand le bâtiment est utilisé. Cette variable est bien présente dans les autres structures mais elle est inactive. Le but principal de ces bâtiments est de produire une ressource, ils ne sont pas uniquement décoratifs et il ne doit surtout pas passer par la tête du joueur qu’il peut s’en débarrasser. Il y a trois types de ressource dans le jeu à l’heure actuelle, la nourriture, les places de logement et l’énergie. La nourriture et les places de vies sont pour les personnages et l’énergie sont réservés



Figure 7: L'interface de l'inventaire

aux bâtiments.

Pour les récupérer, l'inventaire possède une liste de tous les bâtiments présents sur la carte qui produisent de ressources. Cette liste est initialisée au début de la partie et n'évolue pas. Les bâtiments ne pouvant pas être supprimés ni ajoutés. Toute les secondes, il va parcourir cette liste, appeler les scripts va leur demander leurs productions de ressource. Pour l'énergie, cette ressource ne s'accumule, toute les secondes, elle est remise à son seuil minimal (de 5, cette caractéristique sera expliquée plus tard) et on ajoute la production des différents bâtiments qui produisent de l'énergie. Pour les habitations c'est le même système qui est utilisé, la valeur est remise à zéro au début du compteur. Le système change légèrement pour la nourriture qui peut elle être accumulée au cours du temps. (Contrairement à l'électricité). Ensuite, sur cette dernière, la consommation est appliquée. L'inventaire prend la liste des personnages actifs et retire à la réserve de nourriture la consommation. Tant que le nombre de structure ne change pas, ni le nombre de personnages alors l'évolution de la quantité de nourriture va rester constante. Le joueur au début est doté d'une petite réserve qui lui permet de tenir le temps d'améliorer la première ferme et d'être ainsi positif dans le ratio consommation production.

L'utilisation de l'énergie et des places de libre est différente car ces deux dernières énergies ne peuvent pas être stockées. (On peut stocker des lits mais à la fin il manque toujours de la place dans la maison). Pour l'énergie, le jeu va juste vérifier que la différence entre la consommation et la production n'est ni nulle ni négative et va alors permettre l'amélioration du bâtiment. C'est pour cette raison que le seuil minimal n'est pas nul car le joueur serait dans l'incapacité de se développer et donc de gagner. Il en va de même pour les maisons. Le jeu commence avec trois personnages, il est donc évident que ces derniers ont déjà des lieux pour vivre. Dans le jeu, le seuil initial est donné par le quartier général qui va permettre de les héberger même si sa fonctionne principale est différente.

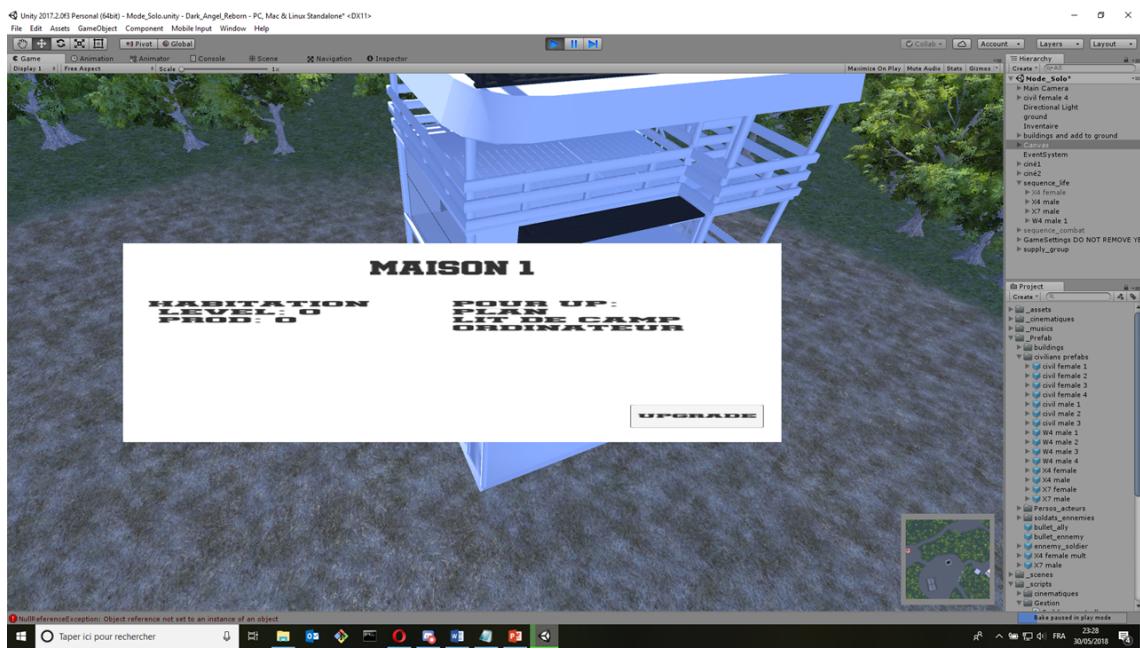


Figure 8: L’interface d’une maison

3.3 Les assignations de poste

Bien que le QG soit la base de vie des premiers colons est une structure de commandement. Elle sert principalement à gérer les ressources humaines de la base. Les personnages autour du joueur ne servent pas qu’à la décoration, elles servent dans les différentes structures disponibles sur la carte. Principalement dans les fermes et la centrale. Il est aussi possible d’employer des civils comme infirmiers dans les tentes prévues à cet effet. Sans ces êtres de chair (et d’octet) les structures sont incapables de produire des ressources (exception faite des maisons qui ne nécessitent que des améliorations). La production de nourriture au départ du jeu est nulle et le joueur vie sur les réserves fournies au départ du jeu. Pour les ressources d’énergie, il est encore possible de changer mais à l’heure actuelle la production initiale est faible mais présente. Il est cependant fortement conseillé d’ajouter rapidement des personnages afin d’augmenter fortement la production. Le joueur est aussi en capacité de recruter de nouveaux colons quand la base est suffisamment apte à les accueillir (assez de place et de nourriture). Le choix des personnages qui arrivent ne sont pas ceux du joueur mais sont imposés. Ce dernier pourrait vouloir recruter tous les meilleurs au premier moment du jeu et laisserait les civils de côté.

Pour assigner les personnages, le script du QG va appeler une fonction. Cette dernière va chercher les objets avec des tags particuliers, les civils et les soldats X7 qui sont dépourvus de capacités supérieures dans leur domaine.

Il parcourra les listes et trouvera le personnage qui n’a pas de rôle attribué et il lui donnera le poste. Dans le cas des unités spécialisées le script va chercher un seul tag celui du spécial puis il va parcourir la liste des objets trouvés avec une boucle while et s’il y un personnage non affecté, il se verra attribuer le rôle. Les fonctions sont assez simples dans leur écriture, les conditions d’arrêt des boucles sont identiques à celles étudiée en cours pour les algorithmes de recherche linéaire. Il suffit de traverser la liste et de s’arrêter quand l’objet est trouvé. La ligne de code est cependant relativement longue car il faut aller chercher un paramètre d’un autre script attaché à un objet. La fonction GetCompoment est très utilisée dans

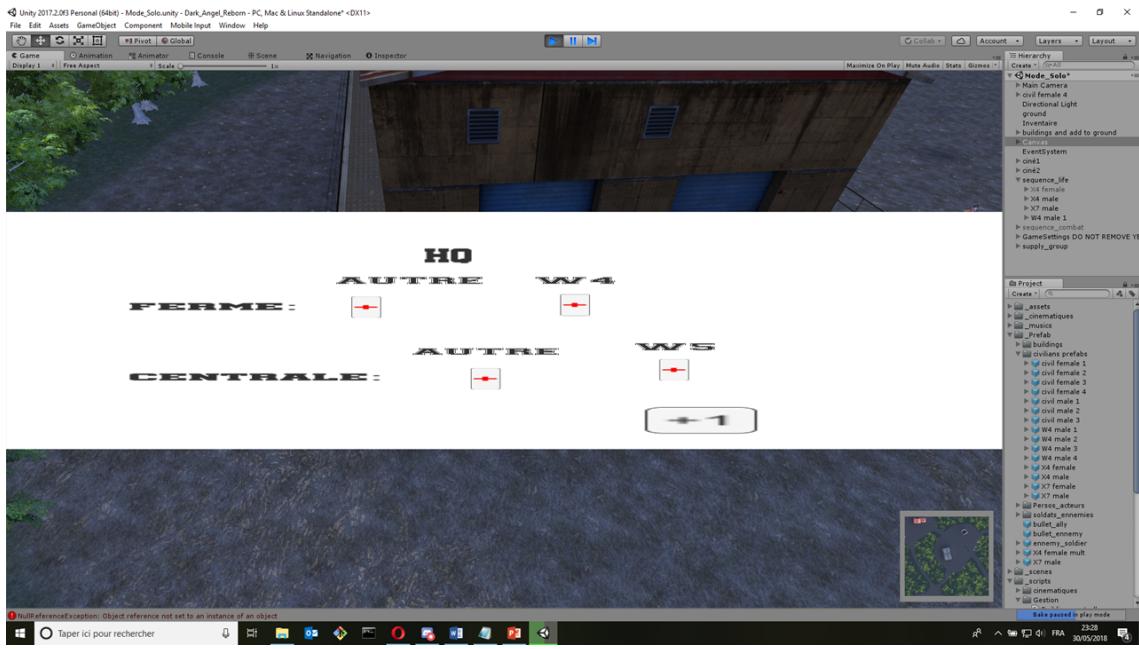


Figure 9: L'interface du QG

ce projet. Pour leur retirer un poste, cela est très simple, le joueur sélectionne déjà le personnage avec les flèches directionnelles. Il ne reste plus qu'à remettre à zéro le nom de leur employeur (comme dans le code). Au départ, l'objectif était d'attacher les personnages via le script des bâtiments mais un bug sur un ajout avec des listes a obligé à reconsidérer cette option. Les personnages avaient une méchante tendance à disparaître de la liste très rapidement de manière incompréhensible. Actuellement le bâtiment de fonction du personnage se trouve attaché à ce dernier et il ne peut plus s'en défaire sans l'autorisation de son maître, le joueur. Il est important de noter que le player principal ne peut pas être assigné à un poste car il doit rechercher les caisses de ravitaillement tout autour de la carte

3.4 Les issues de la gestion

La partie du mode solo présente certaines issues plus ou moins favorables au joueur.

- Dans l'hypothèse où le joueur aurait réussi à garder tous ses personnages en vie et que ses structures sont bien améliorées, qu'il n'y a pas de blesser ni d'inactifs et enfin que toutes les caisses d'amélioration ont été utilisées alors, il va débloquer le dernier personnage du jeu, Alex. C'est l'héroïne de la série qui vivait à Los Angeles. Cette dernière n'a pas vocation à rester mais elle passera dans le jeu pour apporter des nouvelles de la ville à l'origine de tout. Pour donner suite à ce recrutement, elle va servir quelque temps dans le jeu de manière purement décorative et probablement pour les phases de combat qu'elle annonce. Puis elle repartira pour trouver d'autre colonies, ramener du monde dans la nouvelle cité. Cependant son arrivé va causer une attaque de la base et elle va devoir prendre les armes pour se défendre. Cette issue est celle qui amènera à la fin du jeu et à la victoire du joueur.
- L'autre possibilité relativement probable est que le joueur va être trop lent pour mettre en place ses structures de production t donc que ses personnages vont

commencer un à un à mourir de faim. Normalement le jeu va empêcher cette possibilité de toute ses capacités mais si le joueur n'applique pas de producteur à l'une de ses usines de nourriture alors il verra ses stocks s'amoindrir au point de disparaître complètement. Tous les colons seront alors à l'infirmerie et sans possibilité d'en sortir car les quantités de nourriture produite seront toujours nulles. Le joueur doit cependant viser cette possibilité car elle est peu probable d'arriver, le jeu étant assez généreux sur les ressources produites. Il ne doit cependant pas prendre à la légère le risque de mourir de faim.

4 Le gameplay Combat

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 puissent 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 apparaît 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'arrivée. 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     void OnCollisionEnter (Collision col){
42         if(col.gameObject.name == "Player")
43         {
44             DestroyProjectile();
45         }
46     }
47 }
```

Figure 10: le script du projectile

5 Le réseau

Cette partie présente tous les éléments qui utilisent internet. C'est-a-dire le site internet, le multijoueur, le partage Twitter et la communication sur plusieurs réseaux sociaux.

5.1 Le multijoueur : avancement

Lors de cette partie du projet, j'avais deux objectifs principaux : mettre en place le combat en multijoueur, car bien qu'en place en solo, il n'a pas été prévu pour le multijoueur. Le deuxième objectif était de pouvoir rajouter des personnages, que l'on puisse contrôler de la même façon qu'en solo. Mais aussi d'implémenter un système de victoire/défaite que l'on peut partager sur Twitter.

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais n'avait pas de combat ni de système de victoire/défaite ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. J'ai du totalement recréer les scripts de combat car pas adaptés au multijoueur, une fois que l'équipe adverse est décimée la victoire est remportée. Pour le partage de victoire sur Twitter j'ai utilisé tweetinvi qui utilise l'API de Twitter.

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 seule caméra dès le lancement du multi. Effectivement les autres joueurs se fichent de votre caméra, on peut

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.

5.2 La communication : avancement

Lors de cette dernière 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).

5.3 Ce que j'ai fais jusque ici

Lors de cette dernière 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).

5.3.1 Le multijoueur

Lors de cette partie du projet, j'avais deux objectifs principaux : mettre en place le combat en multijoueur, car bien qu'en place en solo, il n'a pas été prévu pour le multijoueur. Le deuxième objectif était de pouvoir rajouter des personnages, que l'on puisse contrôler de la même façon qu'en solo. Mais aussi d'implémenter un système de victoire/defaite que l'on peut partager sur Twitter.

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais n'avait pas de combat ni de système de victoire/defaite ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. J'ai du totalement recréer les scripts de combat car pas adaptés au multijoueur, une fois que l'équipe adverse est décimée la victoire est remportée. Pour le partage de victoire sur Twitter j'ai utilisé tweetinvi qui utilise l'API de Twitter.

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 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.

5.3.2 Le site

Lors de cette partie du projet, j'avais deux objectifs principaux : mettre en place le combat en multijoueur, car bien qu'en place en solo, il n'a pas été prévu pour le multijoueur. Le deuxième objectif était de pouvoir rajouter des personnages, que l'on puisse contrôler de la même façon qu'en solo. Mais aussi d'implémenter un système de victoire/defaite que l'on peut partager sur Twitter.

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais n'avait pas de combat ni de système de victoire/defaite ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. J'ai du totalement recréer les scripts de combat car pas adaptés au multijoueur, une fois que l'équipe adverse est décimée la victoire est remportée. Pour le partage de victoire sur tzitter j'ai utilisé tweetinvi qui utilise l'API de twitter.

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 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.

5.3.3 Communication

Lors de cette partie du projet, j'avais deux objectifs principaux : mettre en place le combat en multijoueur, car bien qu'en place en solo, il n'a pas été prévu pour le multijoueur. Le deuxième objectif était de pouvoir rajouter des personnages, que l'on puisse contrôler de la même façon qu'en solo. Mais aussi d'implémenter un système de victoire/defaite que l'on peut partager sur Twitter.

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais n'avait pas de combat ni de système de victoire/defaite ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. J'ai du totalement

recreer les scripts de combat car pas adaptes au multijoueur, une fois que l'équipe adverse est decimee la victoire est remportee. Pour le partage de victoire sur tzitter j'ai utilise tweetinvi qui utilise l'API de twitter.

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 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.

5.3.4 Extra

Lors de cette partie du projet, j'avais deux objectifs principaux : mettre en place le combat en multijoueur, car bien qu'en place en solo, il n'a pas été prévu pour le multijoueur. Le deuxième objectif était de pouvoir rajouter des personnages, que l'on puisse contrôler de la même façon qu'en solo. Mais aussi d'implémenter un système de victoire/defaite que l'on peut partager sur Twitter.

Pour ce qui est du multijoueur en lui-même : Il était déjà fonctionnel, rudimentaire mais fonctionnel mais n'avait pas de combat ni de système de victoire/defaite ce qui n'est pas l'idéal pour un jeu multijoueur de nos jours. J'ai du totalement recreer les scripts de combat car pas adaptes au multijoueur, une fois que l'équipe adverse est decimee la victoire est remportee. Pour le partage de victoire sur tzitter j'ai utilise tweetinvi qui utilise l'API de twitter.

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 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.

6 Conclusion

Ainsi donc ce dossier est sur le point de se conclure, il aura présenté notre jeu, Dark Angel Reborn, en plus de son équipe la team TEMP-ète. Ce dossier avait pour objectif de vous présenter notre jeu et ses objectifs. Certains points peuvent avoir été rapidement évoqués, il vous sera possible de nous interroger lors de la soutenance et l'écriture de ce dossier a essayé d'alterner entre une partie de présentation de l'idée et de la réalisation dans le jeu ainsi que d'une partie plus technique.

Le jeu Le jeu se décompose en deux parties, la campagne solo et la partie multi-joueur. La partie solo présente un Gameplay de gestion, qui implique de nombreux script et un nombre important d'interaction entre le jeu et le joueur. Ce dernier doit monter une stratégie pour faire perdurer sa colonie. Les obstacles seront importants, le temps, les ressources et les personnages qui consommeront plus que de raison. Pour la présentation la progression sera rapide et le jeu rapidement terminé mais il est tout à fait possible de rajouter de niveaux aux bâtiments et des interactions nouvelles. A partir d'un certain moment, la colonie tournera toute seule et les combats pourront commencer. Ces derniers relanceront l'ardeur et la stratégie du joueur après une phase plus lente de découverte et d'expansion. Si le joueur n'aime pas jouer seul, il aura l'option de rejoindre une salle de jeu et d'attendre un autre joueur. Ces deux derniers lanceront une bataille qui les opposera, si la partie de gestion est absente c'est pour augmenter rapidement le rythme du jeu. Ce jeu ne serait pas intéressant si on ne citait pas rapidement ses graphismes qui se veulent détaillés et proche de la réalité. A l'origine du projet, il était même question de reproduire toute une zone présente dans la réalité. Ce projet a cependant été mis de côté car le terrain ne correspondait pas aux attentes.

Les objectifs Pour ce qui est de la prise en main de Unity, il reste de nombreux points d'ombre mais le concept global a été compris. Il est facile de penser à des objets qui disparaissent de certaines listes ou d'autre qui se mettent à produire un son trop fort et inapproprié à la scène. Cependant, le projet est relativement propre d'un point de vue arborescence et même dans la scène de campagne solo, il est facile de se repérer pour changer un ordre ou un personnage. Nous pouvons donc dire que cet objectif a été globalement réussi.

Les possibles ouvertures Le jeu à l'heure actuel est très court. Les bâtiments ont au maximum 3 niveaux d'upgrade et ne requièrent pas tellement d'objets spécifiques pour permettre leur évolution. Les personnages peuvent arriver si deux conditions sont réunies. Que le nombre de place et le ratio de nourriture soient suffisamment élevés. Si le jeu continue d'évoluer on peut imaginer que le nombre de critère peut aller en nombre croissant, que des missions extérieures à la base pourraient être lancées, qu'il y aurait plus de métier et plus de structures. A l'heure actuel, les objets 3D sont tirés d'internet, l'un des points de progressions pourrait être de les faire nous-même. Ensuite, il n'y a pas beaucoup de dialogue, et les animations ne prennent pas en compte les armes des joueurs. Sur ce point il pourrait y avoir un travail important qui pourrait être réalisé. Le travail qui a été réalisé pose les bases du jeu, pour l'améliorer, il faudrait revoir les codes des bâtiments et de faire une classe mère qui aurait en héritage chaque édifice. Le travail est important mais les ouvriers peu nombreux.

Nous sommes fières de notre travail. Et nous espérons qu'il vous aura convaincu.

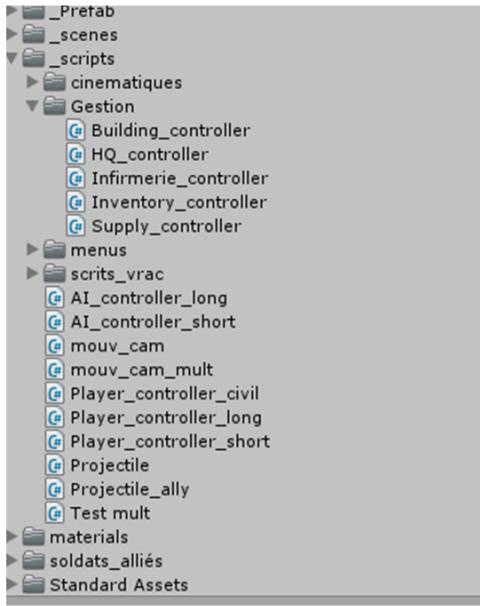


Figure 11: L’architexture des scripts

7 Sources

7.1 modèles 3D

7.1.1 les batiments

- tower house : <https://free3d.com/3d-model/tower-house-design-48197.html> par 3dhaupt
- ghetto : <https://free3d.com/3d-model/futuristic-ghetto-building-23385.html> par 3dhaupt
- tentes : <https://free3d.com/3d-model/tent-85111.html> par : azlyirnizam
- maison: <http://www.turbosquid.com/Search/Artists/TomaszCGB>
- maison: <https://free3d.com/3d-model/house-33742.html>
- caisse ravitaillement: <https://free3d.com/3d-model/supplies-6321.html>
- hunter cabin: <https://free3d.com/3d-model/house-10669.html>

7.1.2 les armes

- Beretta <https://free3d.com/3d-model/rpd-police-beretta-96901.html> par : steve45
- Mk 16 : <https://free3d.com/3d-model/mk-16-51855.html> par : jasonowen
- Shotgun : <https://free3d.com/3d-model/shotgun-29957.html> Soumis par y2k99
- M200 : <https://free3d.com/3d-model/sniper-rifle-m200-36415.html> par : ah-metsalih
- m60e4 : <https://free3d.com/3d-model/m60e4-79415.html> par : ysup12

7.2 Son et musique

- musique des cinématiques: Aggressive War Epic Music Collection! Most Powerful Military soundtracks 2017 [retoché]
- musique principale du jeu: compass de two step from hell

7.3 les visuels:

7.3.1 le logo

- l'ange : <http://darkdreams.centerblog.net/voir-photo?u=http://darkdreams.d.a.pic.centerblog.net/2017/01/logo-ange-100x100px.jpg>
- le drapeau: <https://www.crwflags.com/fotw/images/f/fic-mut.gif>

7.3.2 les polices

- police de texte Varsity: <https://www.dafont.com/fr/varsity-2.font>
- police de texte Soccer: https://www.dafont.com/fr/search.php?q=soccer_league

7.4 les sources d'inspiration

- ToutApprendre: <https://www.youtube.com/user/ToutApprendre>
- Unity Pour Les Nul: <https://www.youtube.com/channel/UCuU8cONIgZ182KheI1s6HqQ>
- Brett: <https://www.youtube.com/channel/UCHzmjxVdrqadOw6lo1Cy5cA>

8 Annexe



Figure 12: La caisse de ravitaillement et sa fumée

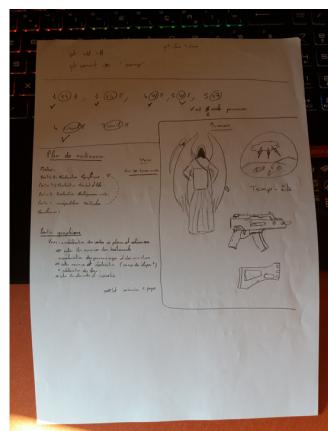


Figure 13: les croquis du logo



Figure 14: ensemble des notes manuscrites du projet (1)

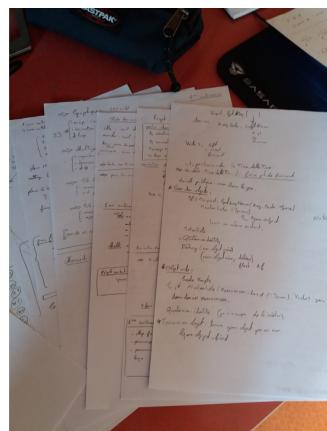


Figure 15: ensemble des notes manuscrites du projet (2)