Rapport De Projet - Year Zero

Bearth Studio

17 mai 2019

timothee.ribes nicola.brankovic

enguerrand.vie axel.ribon

Table des matières

1 Introduction

1.1 Présentation du Groupe

Bearth Studio est un studio de développement de jeux vidéos né le 16 décembre 2018. Composé de quatre membres, sa seule prétention est de mener à bien son premier projet, constitué de son premier jeu : *Year Zero*.
Issus de milieux divers et variés, ses membres possèdent des cultures et éducations différentes. Au sein de *Bearth Studio*, nous pensons que nos différences contribuent à une plus grande ambition créatrice et à une plus large vision d'ensemble de notre projet, de sorte qu'il s'en trouve enrichi. Nous parviendrons alors à résoudre les défis techniques et artistiques qui se présenteront à nous lors du développement de *Year Zero*. N'ayant actuellement qu'un seul projet en cours, nous restons concentrés sur celui-ci mais il est toujours possible qu'à l'avenir nous retravaillions ensemble sur d'autres projets pour que *Bearth Studio* prenne de l'ampleur et continue à présenter des jeux de qualité, témoins de l'ambition commune que ses membres partagent.

1.2 Nom du Groupe

Bearth Studio provient de la contraction de *beyond Earth*, "par delà la Terre" et *birth*, la "naissance". On comprend alors la signification "la naissance, par delà la Terre". Intimement lié au nom de son premier jeu *Year Zero*, *Bearth Studio* représente alors la naissance ou la renaissance de l'humanité et ce, par delà les limites connues de la Terre. Voulant nommer notre groupe en rapport au nom et au thème de notre premier jeu, trouver le nom de ce dernier nous a orienté afin de déterminer celui de notre groupe. C'est pourquoi l'un ne trouve tout son sens qu'avec l'autre. Le cadre était alors posé pour *Year Zero*, présentant une humanité ayant quitté la Terre, cherchant à vivre plutôt qu'à survivre. Ce départ est donc l'année zéro.
Nous avons tenu à ce que le nom de notre groupe soit tiré de notre premier jeu afin que ce soit lui qui définisse les premiers code de notre studio ainsi que ses particularités. Nous ne voulions pas, au contraire, trouver un nom que nous seul comprendrions et qui ne toucherait et ne marquerait l'esprit de personne.

1.3 Logos de Bearth Studio et de Year Zero

Le logo de *Bearth Studio* est composé de trois parties distinctes. La première est celle que nous apercevons dès le premier regard : une planète aux design stylisé, entourée d'anneaux comme par exemple la planète Saturne. On voit alors que s'échappe une fusée de l'orbite de cette planète laissant derrière

elle une traînée de couleur. Le nom du studio apparaît en dessous dans une police d'aspect numérique et spatial et futuriste. L'image que l'on se fait du logo, présentant notre groupe, est simple et accrocheuse de sorte que le joueur se souvienne du logo du studio. Le tout ayant pour caractéristique d'être épuré et moderne afin de montrer le thème principal qui est celui de l'espace, démontrant la continuelle propension du studio à être tourné vers l'avenir.

En ce qui concerne le logo de *Year Zero*, il s'agit d'un cercle contenant en son centre la lettre "Y" stylisé dans une police un peu futuriste qui restera celle de notre jeu. Le cercle représente alors le "0" de *Year Zero* avec le "Y". Les couleurs utilisées sont dans la même gamme que celles du logo de *Bearth Studio*. Un aspect rouille, métallisé a été donné à la lettre "Y" pour correspondre à l'idée d'un futur comme un âge du métal. L'aspect un peu sale ainsi donné peut faire penser à l'ancienne Terre quittée par ses habitants dans l'histoire du jeu.

FIGURE 1 – Logo de Bearth Studio

FIGURE 2 – Logo de Year Zero

1.4 Membres de Bearth Studio

Bearth Studio est composé de quatre membres issus de la classe C1, de la première année du cycle préparatoire intégré de l'école d'ingénieur informatique EPITA.

Chef de projet du groupe, **Timothée Ribes** use de toutes ses compétences afin de créer une vrai cohésion au sein du groupe, d'inspirer et soutenir ses collègues au long du projet. Il est en charge des différents aspects sonores du jeu.

Débordant d'inspiration, mais ne négligeant pas non plus le travail,**Nicola Brankovic** s'occupe des aspects d'écriture sous jacents au jeu et de le faire prendre vie pour de bon dans la mesure où il sera situé dans son contexte, posant ainsi un cadre, une histoire et un univers. C'est en relation avec le travail des tous les autres membres du groupe que Nicola travaille.

Maîtrisant quelques logiciels nécessaires au bon développement de l'aspect graphique et visuel, **Enguerrand Vié** a pour charge la direction de l'aspect artistique du jeu. Devant faire en sorte que chaque aspect du jeu respecte une sorte de charte graphique, des codes couleurs, de formes etc.

Ayant acquis auparavant une certaine expérience avec Unity, **Axel Ribon** aide les autres membres que ce soit du point de vue de la programmation ou dans celui de la gestion du moteur de jeu. Maîtrisant également la programmation orientée objet. Il est, pour ces raisons principalement chargé du code principal du jeu, ce qui constitue le corps du jeu.

2 Présentation Générale

2.1 Présentation

2.1.1 Origine

Avant de choisir le contenu, ou même l'univers de notre jeu, il a fallu déterminer le genre. De nos jours, beaucoup des jeux mondialement connus sont des *FPS*, des *First Personal Shooter*, soit des jeux de tirs ayant une vue à la première personne, voyant à l'écran ce que le personnage joué voit. Après un rapide état de nos connaissances et de nos capacités en terme de développement, nous avons conclu que nous ne ferions pas un *FPS* car demandant beaucoup plus de ressources et de temps pour un même rendu, par rapport à un jeu d'un genre stratégie. Chaque membre de notre groupe étant passionné de jeu de stratégie

en temps réel : *RTS*,*Real Time Strategy*, l'idée d'en faire un nous-même nous est rapidement venue et nous a tous mise d'accord.
Pour ce qui est de l'univers du jeu, constatant l'absence d'un vrai jeu de stratégie se déroulant dans l'espace, nous avons alors décidé de tenter de combler ce vide et de créer le nôtre. Il aurait été alors aisé de reproduire un jeu déjà existant dans le même univers mais nous préférions en recréer un de toute pièce, prenant diverses inspirations mais surtout dans l'optique de faire un jeu à notre image, qui contienne les mécanique que nous voulions, en d'autres termes, que nous créions notre propre jeu.
Nous voulions intégrer une race alien dans notre jeu, en s'inspirant d'insectes comme les abeilles avec leur façon de faire leur ruche et de s'organiser ou encore des fourmis. Le tout étant vu d'une échelle grande. Nous avons aussi pensé à faire de notre terrain de base une sphère de sorte qu'en déplaçant la caméra dans une direction, on reviendrait au bout d'un certains temps au point de départ mais nous avons abandonné cette idée car elle pouvait entraîner chez les joueurs des cinétose, semblable au mal des transport.

2.1.2 Nature

Year Zero est un jeu de stratégie se déroulant dans l'espace et dans lequel plusieurs équipes s'affrontent pour la suprématie de l'univers. On y contrôle des bâtiments qui produisent des unités sur une carte aux limites fixées.
Le moteur de jeu choisi est Unity et le langage utilisé est le CSharp (C). La direction artistique est fantaisiste et non réaliste afin de donner un ton sombre mais non choquant. L'objectif étant de rendre le jeu tout public. L'aspect cartoon a été préféré au réalisme afin de rendre l'ambiance du jeu assez légère mais sans perdre l'aspect stratégie. Cependant nous avons tenu a garder un certains degré de réalisme afin de garder un côté dramatique au vu de l'histoire du jeu.
D'un point de vue sonore le jeu est assez épique avec de grand thèmes entraînant. Mais il jongle avec des thème plus calme et mélancolique pour rappeler la condition désespérée de l'humanité. L'aspect épique étant purement ludique et l'aspect dramatique servant a renforcer les intentions du scénario.

2.1.3 But et intérêts

Une session de Year Zero dure entre 15 et 30 minutes en moyenne. Le jeu mêle stratégie pure ainsi que dynamisme afin de créer une véritable notion de progression durant la partie. Le temps nécessaire à une partie peut sembler long mais il est nécessaire pour laisser le temps aux joueur d'élaborer des stratégies

FIGURE 3 – Aspect cartoon mais sérieux
Endless Space 2

et d'installer sa base.
Nous pensons aussi qu'accélérer les parties rendrait certes le jeu plus accessible mais aussi moins exigeant. Or nous avions la volonté de créer un jeu facile à comprendre mais difficile à maîtriser de sorte que le joueur tire une réelle satisfaction à vaincre ses ennemis.

2.1.4 Year Zero dans l'histoire du jeu vidéo

Year Zero est un jeu de stratégie en temps réel. Le genre existe déjà depuis près de 40 ans avec le jeu *War of Nerves* !. Mais celui ci s'est démocratisé et renouvelé pour trouver sa forme actuelle avec *Warcraft* sorti en 1994. D'autres saga de *RTS* célèbre ont vu le jour comme *Age of Empires* dont la particularité est de pouvoir avancer dans les périodes de l'histoire au fur et à mesure de la partie. On peut encore citer *Age of Mythology* se déroulant comme son nom l'indique dans un contexte mythologique. Plus récemment *Battle for Middle Earth* se déroulant dans l'univers du *Seigneur des Anneaux*. Mais dans l'histoire du *RTS* on retient surtout la saga *StartCraft* pour son aspect compétitif et *Warcraft III* pour avoir ajouté la notion de "Héros" menant quelques années plus tard à la création d'un nouveau genre très populaire qu'est le *MOBA*, *Multiplayer Online Battle Arena*.

2.2 Fonctionnel

2.2.1 Rappel Contextuel

L'humanité a épuisé toute les ressources de la planète Terre la délaissant dans sa pollution. Elle était le théâtre de nombreuses guerres menant à la quasi-extinction de l'humanité. Dans un dernier espoir, les civilisations restantes partirent à la conquêtes de l'univers pour se développer à nouveau. Le point de départ de la conquêtes de l'univers et de la renaissance de l'humanité est appelé *Year Zero*.
Durant la campagne principale le joueur devra effectuer diverses batailles avec des civilisations extra-terrestre pour devenir de plus en plus puissant et affirmer sa suprématie. Ses batailles pourront ressembler au mode de jeu classique avec la gestion de sa base. La campagne pourra aussi être une aventure où le joueur contrôle un régiment de troupes mais sans bâtiments le but étant de survivre jusqu'au bout de la mission. D'autres missions encore demanderont au joueur de résister pendant un certain temps face à des vagues d'ennemies qui attaqueront.

2.2.2 Fonctionnalités du jeu

Le but du joueur est de développer sa civilisation afin de détruire la ou les équipes ennemies. Pour cela il va pouvoir collecter des ressources afin de construire et d'améliorer ses bâtiments, et ainsi former une armée. Le jeu sera décomposé en deux modes de jeu. Le mode solo et le mode multijoueur.

Mode solo :
Campagne - La campagne est découpée en x mission qui se suivent. Une fois que le joueur a terminé une mission il peut passer à la suivante et ainsi de suite jusqu'à arriver à la fin du jeu.
Les différentes missions du jeu proposent des objectifs variés afin de briser une éventuelle monotonie :
- Les missions classiques demandent au joueur de développer sa base ainsi que son armée pour anéantir l'adversaire.
- Les missions de survie sont identiques aux missions classiques si ce n'est qu'il ne faut pas anéantir l'ennemi mais uniquement des vagues de troupes en un temps donné. Le joueur est cloîtré aux environs de sa base et ne pourra attaquer les bases ennemies.
- Les missions aventures fournissent en début de partie un régiment de troupes au joueur qui va devoir user de sa stratégie pour parcourir tout le niveau en gardant au moins une unité en vie.

Partie Rapide - Ce mode correspond au mode multijoueur mais hors ligne. La seule différence étant que seules des intelligences artificiels peuvent être affrontées ici (Voir multijoueur).

Mode multijoueur :
Cœur du jeu, ce mode permet aux joueurs d'essayer leurs stratégies les uns contre les autres. Le seul mode de jeu présent est le mode classique du mode campagne à la différence près que les intelligences artificiels pourront être remplacées par de véritables joueur ou non.

2.2.3 Déroulement d'une partie

Nous décriront une partie multijoueur. Un joueur doit créer la partie depuis le menu du jeu. Il doit décider de la carte de la partie, le nom de la partie ainsi que du nombre maximum de joueurs. Les joueurs voulant rejoindre la partie doivent rentrer le nom de celle-ci dans le menu adéquat pour entrer dans la "salle d'attente".
Dans cette salle d'attente chaque joueur décide de la couleur qui lui est associée, de la race de sa civilisation ainsi que de son équipe. Ensuite ils doivent cocher la case "*Ready*" pour permettre au joueur ayant créé la session de lancer le jeu. Les joueur sont ensuite répartis aléatoirement sur la carte.
Chaque joueur a ensuite une station spatiale en guise de bâtiment principal. Celui-ci permet de produire des constructeurs qui construisent et réparent les bâtiments, mais aussi minent les ressources pour le joueur. La station peut être améliorée pour devenir plus résistante mais aussi et surtout pour pouvoir débloquer de nouveau bâtiments à construire.
Le joueur doit aussi développer son arbre d'amélioration qui va lui permettre, via un coût en ressources de débloquer de nouvelles unités, de les améliorer ainsi que de débloquer diverses amélioration de production sur ses bâtiments.
Les constructeurs peuvent donc miner diverses ressources et notamment sur les astéroïdes présents sur la carte. Ces ressources sont nécessaires à la construction des bâtiments et des unités mais elles ne sont pas inépuisable. A force d'être minés, les astéroïdes vont se détruire forçant le joueur à en trouver d'autres.
Le joueur a juste à sélectionner un constructeurs et lui assigner un astéroïde pour que celui-ci fasse des aller-retours vers la station spatiale la plus proche pour que la ressource de cet astéroïde soit stockée.
Il existe une autre ressource qu'est la nourriture, introuvable dans l'espace, le joueur doit faire construire des fermes spatiale qui élèveront du bétail pour produire de la viande à intervalle de temps régulier.

Enfin le joueur doit faire attention à la population. En effet, chaque unité a une valeur de population, et la somme des valeurs de population de toutes les unités ne peut pas excéder la population maximum de la base.
La population maximum de la base va dépendre du nombre de bâtiments d'hébergement présent. Avec un plafond de 100 qui ne peut être dépassé. D'autres bâtiments sont constructibles et notamment pour produire différents types de troupe.
Chacun de ses bâtiments se voit mettre à disposition une bannière de ralliement, ainsi chaque unité produite se rendra à ce point de ralliement. Pour ce qui est de la défense le joueur a à sa disposition la possibilité de construire des tourelles de défense qui vont automatiquement attaquer les troupes ennemies à proximité. Le joueur peut aussi placer des sentinelles qui l'alerteront de la présence de troupes ennemies. Car en effet, étant dans l'espace le joueur n'a pas la possibilité de construire des remparts, il ne pourra donc pas se réfugier derrière mais doit préparer sa défense pour survivre. D'où la pertinence de bien placer ses sentinelles pour lui donner un maximum de temps avant l'attaque imminente. Les tourelles peuvent ralentir l'ennemi mais rarement vaincre son armée entière.
Les bâtiments ne sont pas équipés de propulseurs et ne peuvent donc pas se déplacer seul. Ils sont donc en orbite autour de la station spatiales.

2.3 Aspects Techniques et Méthodologiques

2.3.1 Moyens matériels

Nous avons utilisé tout une variété de matériels pour développer notre jeu. Ainsi Visual Studio a été utilisé pour la partie code, Blender pour la modélisation 3D, Photoshop pour le design et Git pour le "*versioning*" du projet. Le tout a été assemblé pour construire le jeu avec Unity. Nous avons à notre disposition 4 ordinateurs portables ainsi que les locaux de l'EPITA. Pour nos recherches nous avons utilisés le moteur de recherche Google. L'EPITA nous met à disposition des éditions de Windows 10 ainsi qu'une licence Office 365 qui nous servent pour nous organiser schématiquement sur la construction du jeu.

2.4 Moyens Intellectuels

Pour nous aider durant le développement nous avons accès au *MSDN* pour tous ce qui concerne le langage C. Unity met à disposition une documentation pour son utilisation et nous avons utilisé aussi beaucoup de topic de forum déjà crée et résolu. Ces forums sont majoritairement issues des sites Unity et

StackOverflow. Ces ressources nous permettent de profiter de l'expérience des autres pour apprendre plus vite. Nous avons pu également nous servir de vidéos explicatives sur certains aspects de Unity non maîtrisés ou sur d'autres logiciels que nous avons été amené à utiliser.

2.5 Moyens Économiques

Le jeu n'est pas monétisé d'une quelconque manière. Le tout est gratuit sans aucun contenu téléchargeable additionnel payant ni abonnement ni "*lootbox*". Nous pensons que le succès du jeu seul serait une récompense et si tel est le cas une vitrine publicitaire mettant en avant le studio.
Le jeu de base n'est pas payant pour la simple raison qu'il est le résultat d'un projet scolaire et que nous ne pensons pas que ce type de projet devrait être rémunéré.

3 Découpage du projet

3.1 Programme de l'avancement des tâches

TâcheSoutenance	**Soutenance 1**	**Soutenance 2**	**Soutenance 3**
Graphisme	30	60	100
Son	20	70	100
Menu	60	80	100
Réseau	50	70	100
IA	10	60	100
Mécanique de jeu	60	70	100

3.2 Tableau de répartition des tâches

	Timothée	Nicola	Enguerrand	Axel
Graphisme				
Troupes			X	
Batiments			X	
Menus			X	
Interface			X	
Bande-Annonce			X	
Son				
Bruitages	X			
Musiques	X			
Interfaces				
Menu Principal				X
Menu en jeu				X
Interface Utilisateur				X
Contrôles				X
Réseau				
Lobby				X
En ligne				X
IA				
Bots				X
Automatismes				X
Mécanique de jeu				
Selection				X
Ressources				X
Combat				X
Construction/Production				X
Contenu				
Map			X	
Campagne	X	X		
Histoire		X		
Autres				
Discord			X	
Site		X	X	

3.3 Développement sur les tâches réparties

3.3.1 Timothée

Ne présentant aucune qualité particulière je me consacre au rôle de chef de projet visant à aider mes collègues du mieux possible facilitant ainsi l'entraide la communication et la cohésion qui sont des principes fondamentaux pour un projet de groupe. Toutefois je joue de la guitare et vais donc utiliser mes capacités musicales pour m'occuper de tout ce qui est en rapport au son dans le jeu, c'est-à-dire créer une ambiance propre à *Year Zero* et touchante pour le joueur avec des musiques adaptées, au ton futuriste mais aussi à la tension que crée ce genre de jeu tout du long d'une partie. Dans le cadre de l'apprentissage du C# à EPITA, pouvoir l'appliquer dans un cadre concret de ce jeu dans les sujets me permet de l'approfondir et de découvrir des sujets plus complexes. En effet avec Nicola et Axel nous allons devoir coder des intelligences artificielles suffisamment performantes pour qu'elles puissent donner de la difficulté au joueur seul sans trop le surpasser. Pour finir, une fois le jeu fonctionnel dans sa partie multijoueur, avec Nicola nous devrons scénariser des parties pour en faire une aventure solo qui paraîtrait minimaliste par rapport au éditions professionnelles de ce genre, mais permettra tout de même d'exprimer l'univers propre *Year Zero* développé par les efforts de tout le groupe.

3.3.2 Nicola

Étant un grand consommateur de jeux de stratégie en temps réels(*RTS*) je me suis dédié aux mécaniques de jeu et de l'IA afin que notre jeu soit agréable à jouer et équilibré. Un *RTS* demande souvent d'effectuer plusieurs actions à la fois, permettre aux joueurs d'ordonner à ses bâtiments ou unités de réaliser plusieurs tâches les unes à la suite des autres, rapidement, est un élément clé du gameplay d'un jeu de stratégie. Réfléchir au fonctionnement des mécaniques du jeu et les transcrire en algorithme puis en C# est enrichissant et pousse notre équipe à une entraide constante et à des débats sur les limites de cette automatisation entre Axel, Timothée et moi. M'occupant également de la campagne et de l'histoire de notre jeu, je vais devoir travailler main dans la main avec Enguerrand et Timothée afin de fournir une certaine cohérence avec les musiques, les thèmes et les designs. Notre objectif est de donner une âme et une atmosphère unique à notre jeu afin d'offrir une expérience unique aux joueurs, de les impliquer dans leur session de jeu, et de les pousser à toujours viser la victoire.

3.3.3 Enguerrand

Possédant avant tout des qualités de travail graphique, mon travail dans le projet *Year Zero* sera particulièrement porté sur l'aspect graphique du jeu. Je suis chargé de la totalité de la modélisation 3D en ce qui concerne les modèles 3D des unités et ceux des bâtiments. Aidé par Axel, je devrai également m'occuper d'habiller les interfaces du jeu notamment les menus, les boutons, l'interface dans une partie etc. Le tout afin de créer une ambiance futuriste, spatiale, avec bien évidemment les couleurs qui s'y rapportent. Le même travail sera effectué en parallèle sur les menus de "*lobby*" multijoueur. Je serai aussi chargé d'un point de vu de la programmation, de tout le système de combat de notre jeu, aidé cette fois-ci de Nicola. Il s'agira alors de faire interagir les unités entre elles, de sorte que le combat soit vivant et dynamique. Un aspect qui allie programmation et graphisme est la génération dynamique des environnements, de la carte du jeu. Voulant intégrer un système de génération aléatoire de carte pour que le joueur ne se lasse pas de rejouer une partie de *Year Zero*, je serai chargé également de développer un tel système.

3.3.4 Axel

Mon travail dans l'équipe a été avant tout algorithmique. Je me suis occupé de certaines mécaniques sur lesquelles se sont reposés mes collègues pour faire leur travail. J'ai donc eut à charge de leur fournir un code propre et de pouvoir le leur expliquer. Je pense que la communication est primordiale pour être efficace et productif et mon travail a illustré cette nécessite dans le sens où il a garantie que nous échangions sur nos travaux. De plus, mon travail a compris aussi l'interface avec l'utilisateur et m'a donc invité à me mettre à sa place et à demander leur avis à ceux qui m'entourent. Bien que très abstraites de prime abord, mes tâches m'ont demandé aussi bien des qualités techniques que sociales.
Ainsi j'ai été chargé de l'algorithme de sélection des unités qui est le pilier du jeu dans le sens où tout contenu ajouté nécessite la sélection d'unité pour pouvoir être testé.
J'ai été aussi chargé du développement du code des menus. Cette tâche peut paraître simple au premier abord mais le but d'un développeur est non seulement de rendre le jeu jouable mais aussi de faciliter le travail des autres en codant un système très adaptatif et facile d'utilisation. Le tout a dû être développé de sorte à ce qu'une fois l'algorithme fini il n'y ait plus besoin de toucher au code mais seulement à l'interface et ce de la manière la plus simple et rapide qui soit.
J'ai été aussi chargé de ce qu'on a appelé les automatismes. Il s'agit de légères

notions d'intelligence artificielle chez les unités. On peut par exemple citer le système de patrouille, les réactions des unités lorsque des ennemies approchent ou encore leur faculté à miner et à ramener les minerais jusqu'à la station spatiale. La notion d'intelligence étant trop faible pour être appelé intelligence artificielle et le fait que ces réactions sont totalement dissociées des actions du joueur nous avons préféré dissocier ces deux parties.
J'ai été cependant aussi chargé de l'intelligence artificielle des joueurs contrôlé par l'ordinateur. Ce fut ma tâche la plus difficile car je n'avais aucune expérience dans ce domaine et que l'IA dans un jeu de stratégie est bien plus complexe que dans une majorité d'autres type de jeu.
J'ai développé aussi tout ce qui concerne la production d'unité et de bâtiments, allant de l'affichage du coût de ceux-ci à la production en passant par un mode de placement de bâtiment dans la base, ainsi que le fonctionnement de l'interface associé. C'est une tâche complexe car tous ces systèmes s'imbriquent les uns dans les autres et donc la modification d'un de ces systèmes affecte les autres.
Une autre part de mon travail a été d'implémenter le multijoueur. Et tout ce qu'il implique à savoir les menus pour créér ou rejoindre un serveur ainsi que la salle d'attente ou les joueurs pourront choisir leur race et leur équipe. Mais le multijoueur a surtout été implémenté tout au long du jeu et dans la gestion de la partie. C'est aussi un travail massif au niveau de l'optimisation pour éviter d'éventuelles latences durant la partie.
Pour résumer je me suis occupé du multijoueur, de la sélection d'unité, de l'interface d'un point de vue code, de la construction de sa base, de la gestion des unités ainsi que leur production et leurs capacités.
Enfin je me suis occupé du système de combat.
Je suis avant tout un codeur et non un artiste, c'est pourquoi cet aspect de la création du jeu ne m'est pas attribué.

4 Développement sur les tâches réparties

4.1 Timothée

Mon travail en tant que chef de projet se situe entre les différents domaines que nous utilisons, tout en étant focalisé sur les aspects sonores mais aussi de la partie avec un seul joueur.

4.1.1 Tutoriel

Le tutoriel consiste en une partie scénarisée dans laquelle le joueur doit accomplir les actions demandées pour avancer tout en comprenant et apprenant les mécaniques du jeu. La première difficulté qui se pose dans ce genre d'exercice est de se mettre à la place du joueur et pour cela oublier le plus possible ses connaissances et habitudes par rapport au jeu sur lequel on joue, voir même par rapport au genre tout entier. De même il faut arriver à lui faire comprendre les actions qu'il doit effectuer et pour cela nous avons décider d'afficher des messages et de mettre en pause la partie le temps que le joueur lise et puisse avancer à son rythme avec sa souris. Nous avons choisi de faire un tutoriel simple qui explique seulement les bases concises du jeu afin de laisser une certaine expérience de découverte et de faire quelque choses de très léger qui puisse au besoin être rejoué très vite pour se remémorer les fonctionnements principaux des unités et bâtiments. Du côté technique cela a consisté principalement en une utilisation des outils mis en place pour le fonctionnement du jeu et à la façon de les relier entre eux par un script qui exécute chaque étape à effectuer. Ce qui nécessite de la communication avec Axel pour la compréhension et parfois l'adaptation du code, mais aussi les ajustements graphiques d'Enguerrand qui permettent au jeu de rester toujours cohérent esthétiquement. C'est donc notre capacité à travailler à quatre sur les mêmes éléments qui est mise en jeu à ce moment là.

4.1.2 Site internet

De mon côté je n'ai jamais pris part à la conception d'un site internet, j'ai donc particulièrement travaillé sur l'aspect du rendu et des fonctionnalités avec Nicola qui possédait déjà des compétences dans ce domaine.

4.1.3 Doublage

L'ensemble de musiques et de sons qui seront parties intégrantes du jeu notamment pendant les parties et les différentes interactions du joueur et les actions des unités elles mêmes devront être développées pour la prochaine soutenance en même temps que les graphismes internes du jeu. De plus les unités posséderont des voix qui répondent aux ordres donnés. Par exemple lors d'un ordre d'attaque donné à un vaisseau on entendra "à l'attaque". De plus les textes et dialogues de la campagne seront doublés pour vraiment créer un récit avec une narration correspondant au scénario de Year Zero.

4.1.4 Mission

La mission fait partie de l'expérience solo que propose Year Zero et se place dans la continuité du tutoriel qui doit être terminé pour y accéder, et va présenter la mécanique la plus attractive dans un jeu de stratégie en temps réel, les combats dont le fonctionnement et l'implémentation sont détaillés dans le travail d'Axel. L'ennemi ne possède donc pas encore de bâtiments économiques ou de production, et le joueur peut se concentrer sur le simple fait de se débarrasser des unités qui vont attaquer sa base en suivant les instructions introduites par la narrateur. D'autre part, un travail d'ajustement devra se faire avec les rééquilibrages du jeu pour que cette partie reste assez simple à exécuter pour le joueur novice. Enfin cette mission a aussi pour but d'introduire le scénario en effectuant le premier contact entre lui et l'antagoniste qui envoie des troupes pour le détruire. De plus, un autre mode qui est sans fin à été ajouté, c'est un mode que nous avions prévu d'ajouter et qui est similaire à cette mission dans lequel le joueur va donc encore une fois faire face à des vagues qui ne lui laisserons pas de repos et se répéteront à l'infini en étant toujours plus fortes pour constituer un challenge pour atteindre les vagues les plus avancées.

4.1.5 Sons

En parallèle il a fallu continuer à développer l'ambiance sonore du jeu, déjà pour les voix des personnages qui arrivent au fur et à mesure de l'histoire, ce qui nous a forcé à faire du montage audio à l'aide du logiciel Audacity et pour lesquelles nous avons choisi d'utiliser nos voix malgré notre absence d'expérience dans le domaine du doublage.Nous avons réalisé la diversité des métiers impliqués dans un jeu. Nous espérons donc pouvoir être sérieux sur ce domaine mais laissant tout de même notre trace à travers nos voix aussi modifiées qu'elles soient. On prend donc soin de préparer les répliques avec une certaine idée du ton qu'elle doivent avoir, puis on effectue des essais jusqu'à avoir une version pertinente dont on enlève ensuite le bruit et à laquelle on ajoute des effets de hauteur pour coller au personnage qui doit être incarné. Aussi, je me suis assuré que l'aspect musical soit fonctionnel dans le jeu, c'est à dire qu'on puisse lancer les musiques en fonction du moment, qu'elles ne se superposent pas, ni ne se redémarrent ou se coupent en fonction des changements d'environnement comme rejoindre une partie ou changer de menu. De plus pour que la musique dans ce jeu soit une partie intégrante de l'expérience de jeu j'ai pensé à, mieux que faire une liste de sons aléatoire qui se lisent au fur et à mesure d'une partie, allier leur ambiance à l'état de la partie. En effet des musiques plutôt calmes seront jouées en début de partie, puis au fur et à mesure que le joueur débloque

des unités ou des technologies, les sons seront de plus en plus énergiques ayant pour but d'aboutir à un cadre épique pour les grandes batailles de fin de partie qui mettront en jeu beaucoup d'unités.

4.1.6 Equilibrage

L'équilibre d'un jeu de stratégie, même dans sa version la moins poussée est un concept principal, de façon à ce que les différentes unités forment un tout. Il permet que chaque vaisseau de notre jeu ait sa place et son utilité soit en ayant des statistiques avantageuses pour un coût plus élevé, ou plus faibles mais avec des sorts plus puissants. Atteindre un équilibre parfait nécessiterait une analyse de données sur un nombre énorme de parties et de joueurs, c'est pourquoi nous nous sommes appuyés sur des séances d'essais personnels du groupe qui devront être poussées plus loin pour la prochaine soutenance afin de continuer à affiner l'équilibre et une bonne expérience intéressante. Pour cela nous avons créé un tableau rassemblant toutes les unités auxquelles nous avons assignées les diverses statistiques : vitesse d'attaque, dégâts, portée, points de vie, vitesse et leur coût dans les trois ressources du jeu. A cela nous avons ajouté des coefficients à chacune de ses valeurs et les avons regroupés dans une valeur propre à l'unité. Notre but était alors que toute les unités aient la même valeur tout en modulant leur différentes caractéristiques, mais nous avons abouti à quelque chose où les valeurs tendaient à croître en fonction de la fin de partie car nous avions négligé les recherches à faire à travers l'arbre de compétence nécessaire pour obtenir ces unités qui sera lui équilibré pour la prochaine soutenance avec le projet d'avoir un déroulement de partie complet jouable et agréable.

4.2 Nicola

4.2.1 Site internet

N'étant pas spécialisé dans la création de sites Internet, j'ai décidé de me servir de l'outil Wix afin de fournir un résultat propre. J'ai tenu à garder le thème du jeu dans le site internet, j'ai donc réutilisé une des polices dominantes de notre jeu et l'image de fond du menu. J'ai ensuite suivi le plan proposé dans le Dossier Projet, et donc incorporé les liens de téléchargement du Cahier des charges, du rapport de soutenance et de la version "lite" du jeu, et une présentation complète de membres de Bearth Studio. Cependant, mes capacités en tant que graphiste et mon bon goût n'étant que subjectif et limité, j'ai laissé ensuite place à Enguerrand qui s'est chargé de finaliser le site web.

4.2.2 Tutoriel

Le tutoriel à pour but de présenter les mécaniques de bases du jeu, il pose l'ambiance du jeu, et donne les principales clés de victoire d'un RTS. Pendant celui-ci, le joueur se verra présenter comment sélectionner les unités/bâtiments mais également les bâtiments de base et leurs utilités, l'utilisation des Builder qui est l'unité de construction/récolte, les unités de combat et comment les créer, comment récolter des ressources, comment les utiliser et enfin le fonctionnement du SkillTree, élément original au centre du gameplay de Year Zero. Afin de forcer le joueur à écouter les consignes du tutoriel, nous avons du limiter les actions que celui-ci peut réaliser, et vérifier que les demandes du tutoriel sont satisfaites avant de continuer. Le tutoriel, étant l'une des premières expérience du jeu que nous avons pu avoir, cela à permis de faire ressortir certains bugs ou mécaniques de jeu incomplètes voir absentes, mais également certaines implémentations qui seront nécessaire pour continuer la campagne.

4.2.3 Mode solo

Les bases du mode solo ayant été implanté pour la soutenance deux, nous avons décidé, avec Timothé de fournir un dénouement au Scénario de Year Zero. La mission se veut cette fois plus longue et complète et sera encore une fois doublé. Permettre à Year Zero d'avoir un scénario complet est une brique essentielle, parmi tout celles qui lui permettent d'être au niveau des standards du jeu vidéo actuel. Dans les nouvelles missions implantés, cette fois, le joueur devra se battre contre l'intelligence artificielle de Year Zero, et aura pour but de détruire tout les bâtiments de l'ennemi. Le tout est scénarisé et offre des aides et indications au joueur afin de le faire battre notre très competante IA.

Le mode solo en général à par ailleurs été revu, notamment au niveaux de l'équilibrage, afin que la difficulté suivent la courbe de progression du joueur.

4.2.4 Scénario

Suite à des concertations avec le groupe le Scenario de Year Zero, présenté à lors de la premiere soutenance à été modifié. Celui-ci peut se retrouver dans la vidéo promotionnelle du jeu. mais voici un synopsis :

Après la lente et inéluctable destruction de la terre par l'homme, celui-ci a été forcé de s'orienter vers les cieux pour assurer sa survie. Hélas l'espace est loin d'être inhabité, et l'humanité se voit forcer de relever de nombreux défis. Parmi eux, une race alien ayant conquis de très nombreux système, extrêmement agressive, elle peut s'adapter à tous les environnements, mais le plus effrayant ? Elle peut prendre possession des corps de vos amis, de vos frères d'armes, de

vos chefs, de votre famille... Les combats sont quasi permanant et l'homme n'a nulle part ou s'installer. L'humanité est proche de l'extinction, et attend désespérément son messie.

4.3 Enguerrand

4.3.1 Création des logos

Responsable de la partie visuelle du jeu, je me suis penché en premier, au moment de rédiger le cahier des charges, sur la conception d'un logo pour le jeu, ainsi que pour le groupe. Une sorte de studio de développement et d'édition de jeu vidéo. Déjà présentés lors du rendu du cahier des charges, les logos de Year Zero et de *Bearth Studio* contiennent majoritairement les couleurs rouge, bleu et violet dans diverses teintes. Voulant donner un aspect spatial pour se rapporter au thème principal du jeu, les logos sont devenus référence dans toute la conception future dans ce qui concerne le design du jeu.

4.3.2 Conception de l'identité graphique

Hormis les logos, qui ne sont qu'une pièce moindrement visible que l'interface du jeu par exemple, il a fallu déterminer les lignes directrices de la direction artistique de Year Zero afin de conserver les mêmes couleurs, formes etc. Le tout, dans la prétention de vouloir créer une atmosphère entraînante et immersive. C'est donc dans des tons de couleurs chauds, se rapprochant du rouge, du violet et du bleu que Year Zero prendra forme. La police que nous utiliseront majoritairement est *Orbitron Black* pour la plupart des textes et pour certains titres nous utiliserons Space Age.

FIGURE 4 – Police Orbitron Black

FIGURE 5 – Police Space Age

4.3.3 Menu Principal

C'est vers le menu principal que je me suis penché en premier. Au menu déjà existant, constitué de boutons sans texture, et du fond par défaut de Unity, j'ai ajouté un fond qui est une image dans l'espace où l'on voit des nébuleuses. J'ai ajouté en bas a gauche et à droite respectivement les logos de Year Zero et de Bearth Studio. Enfin j'ai créer sur Photoshop le logo titre Year Zero que j'ai importé dans Unity afin de le mettre au dessus des boutons.

N'ayant pas d'idées précises de comment j'allais designer les boutons, j'ai tenté quelques versions avec diverses couleurs, toujours en créant une image de bouton sur Photoshop qui allaient se superposer au bouton Unity.

Au final, j'ai créé une nouvelle texture pour les boutons que j'ai importé. Il s'est avéré qu'elle collait bien au fond et avec le reste, ainsi on l'a gardée.

FIGURE 6 – Texture du bouton

J'ai décliné cette même texture en plusieurs versions afin de s'intégrer au mieux dans toutes les scènes de notre projet Unity et dans toutes les situations ou nous allions en avoir besoin.

FIGURE 7 – Bouton pour une barre de remplissage

FIGURE 8 – Bouton carré aux bords arrondis

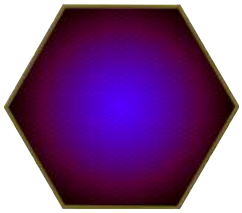


FIGURE 9 – Bouton en forme d'hexagone

FIGURE 10 – Bouton rectangle aux bords arrondis

Finalement, c'est vers des boutons en forme d'hexagone que je me penche, voulant innover un peu de sorte que nous n'ayons pas des boutons qui semblent être seulement par défaut ceux de Unity. Au lieu de marquer où ces boutons redirigent, je me suis alors dit que c'était probablement plus explicite et aussi plus agréable visuellement de mettre des icônes au lieu du texte. Nous avions alors un menu qui commençait à prendre forme.

J'ai aussi rajouté pour un effet d'immersion au sein du menu, deux couches d'étoiles matérialisées par deux images contenant du transparent et des étoiles. Je leur ai ajouté une animation de sorte qu'elles tournent sur elles-mêmes dans s'arrêter. J'ai aussi créer via un petit script, un effet de parallax (que l'on retrouvera plus tard). La parallax est une façon de rendre de la profondeur et de la distance. Ici, il s'agit de faire suivre les mouvements de la souris dans le menu principal à l'image de fond mais avec un léger coefficient. L'effet est très léger

FIGURE 11 – Menu principal

ici, il n'est que visuel.

4.3.4 Ajouts de ressources graphiques

A l'issue de la première soutenance, je me suis dit qu'il fallait supprimer tout ce qui, jusqu'alors était *par défaut*. C'est dans cette démarche que j'ai eu l'idée de concevoir un curseur qui viendrait remplacer celui que *Windows* possède.
Il fallait alors que ce curseur puisse s'intégrer dans la charte graphique du jeu sans pour autant perdre en visibilité. Après plusieurs essais non concluants, souvent parce que les couleurs choisies le rendaient moins visible, je suis enfin parvenu à la version actuelle qui est intégrée au jeu.

4.3.5 Écran de chargement

Outre la scène du menu principal, il y a une scène qui n'est que rarement visible puisqu'elle se passe très vite. C'est la scène de l'écran de chargement. Elle est simple dans la mesure ou elle n'est constituée que du même fond que le menu principal, ainsi qu'une barre de progression et d'un message qui affiche *LOADING...*

4.3.6 Animations des boutons

Rajouter des animations aux boutons était nécessaire pour créer du dynamisme dans la navigation au sein des menus. L'animation est la même pour tous les boutons du menu principal et de ses sous-menus. Elle consiste au grossissement du bouton. Ainsi quand la souris passe sur le bouton, ce dernier voit sa taille augmenter légèrement.

4.3.7 Écran de jeu

Environnement global

Lorsqu'on arrive dans une partie, on est dans l'espace. Il est nécessaire d'avoir un environnement adapté. Ainsi le sol d'où bougent les unités est englobé dans une *Skybox* ce qui correspond à une sphère dont la texture est une image, ici de l'espace. Le rendu fait en sorte de respecter une sorte d'immersion dans cette sphère et donne l'impression que la profondeur est infini.

Effet de parallax

Lorsqu'on déplace la caméra dans la partie, on n'a d'impression de mouvement que si on voit défiler des unités. C'est pourquoi j'ai rajouté plus de quatre couches d'étoiles en dessous du sol, toutes à des hauteurs différentes de sorte que lorsqu'on se déplace, les étoiles suivent plus ou moins le mouvement en fonction de si elles sont loin du sol ou pas.

Inteface

Le plus gros de mon travail jusqu'à la soutenance présente a été la conception de l'interface dans une partie. Tout a été fait sur Photoshop puis importé dans Unity. J'ai réalisé en une image qui constituera l'interface globale dans le jeu.

FIGURE 12 – Image de l'interface

De la même manière que pour le menu principal, les boutons d'actions des unités ont, au lieu d'un texte, une icône, ainsi que la texture d'un bouton carré (*cf : Figure 7*).

FIGURE 13 – Interface en jeu

On retrouve également les boutons de navigation au dessus de l'écran. La texture qui leur est associée est le bouton rectangulaire à bords arrondis. Le texte a été laissé pour une meilleure compréhension de leur action et la couleur a été mise sur un doré, jaune qui se porte bien au reste de l'interface.
A leur droite, on retrouve un bouton pour l'arbre ce compétence, avec son icône associée. A sa droite également, on retrouve les indicateurs de ressource. A l'origine, ce sont des boutons mais on ne peut pas interagir avec eux, d'où leur couleur plus sombre que les autres. La couleur du texte et des chiffres a aussi été mise sur le doré, jaune.

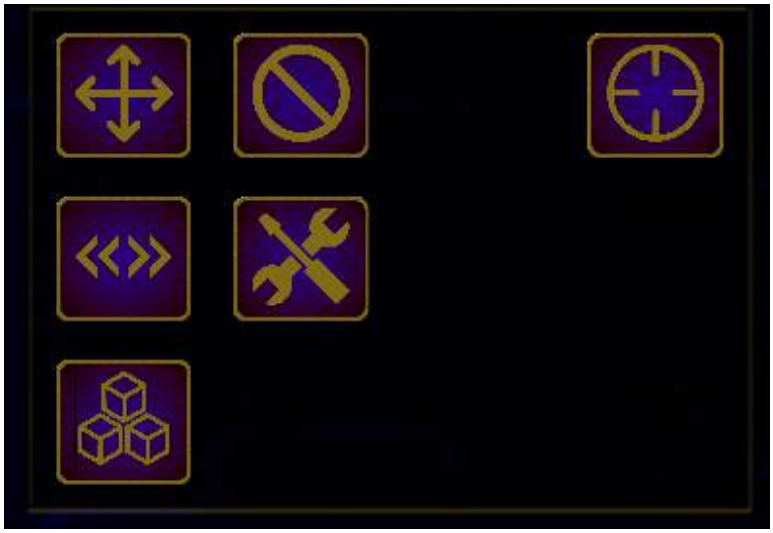


FIGURE 14 – Boutons avec icônes

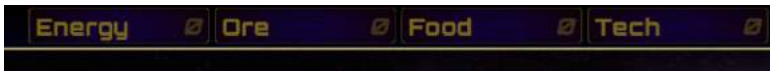


FIGURE 15 – Boutons des ressources

Sont présents également les boutons de la minimap, formés eux aussi de leur texture et d'une icône, comme pour toutes les autres faite par mes soins sur Photoshop. On retrouve aussi le bouton de constructeur inactif constitué de la même manière. Tous ces boutons possèdent la même animation, qui change leur couleur en fonction de leur état. Les quatre états distincts étant : *Highlighted* si la souris est sur le bouton, *Pressed* si la souris a cliqué sur le bouton, *Disabled* si on ne peut pas interagir avec le bouton ainsi que l'état par défaut.

FIGURE 16 – Bouton survolé par la souris

FIGURE 17 – Bouton cliqué

4.3.8 Amélioration de la visibilité

Un jeu dont l'environnement dominant est l'espace comme Year Zero, a souvent des couleurs sombres. Les textes doivent être présentés dans des couleurs vives pour qu'ils soient bien vu de la personne devant l'écran. Ainsi la plupart des messages auront des couleurs comme le rouge ou le jaune-doré. Pour certains ils pourraient être encore plus visible et c'est dans cette optique que je leur ai appliqué un fond noir en dégradé de sorte qu'ils soient encore plus marquants parmi tout ce qui est affiché à l'écran.
Ainsi les messages du tutoriel expliquant comment jouer seront un peu plus visibles ainsi que les messages d'aide sur les boutons d'actions des unités, qui ne comportaient d'ailleurs pas de message d'aide auparavant.

Par ailleurs, certains menus ont étés améliorés avec quelques détails graphiques pour les rendre plus dynamique. Il s'agit de l'écran de connexion, et du *lobby*, la salle d'attente avant que la partie ne se lance en multijoueur.
J'ai aussi appliqué les créations graphiques de Year Zero au menu que l'on peut trouver en jeu, lors d'une partie. On y accède en appuyant sur le raccourcis écrit

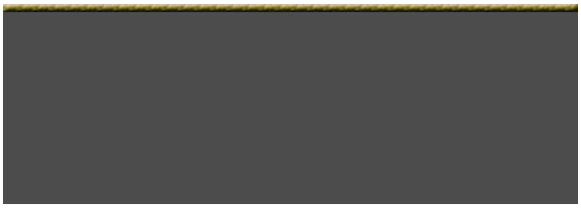


FIGURE 18 – Bandeau de fond pour les textes

FIGURE 19 – Aide pour les actions des unités

sur les boutons de navigation en haut de l'écran ou en appuyant sur la touche *echap*.

FIGURE 20 – Menu principal dans une partie

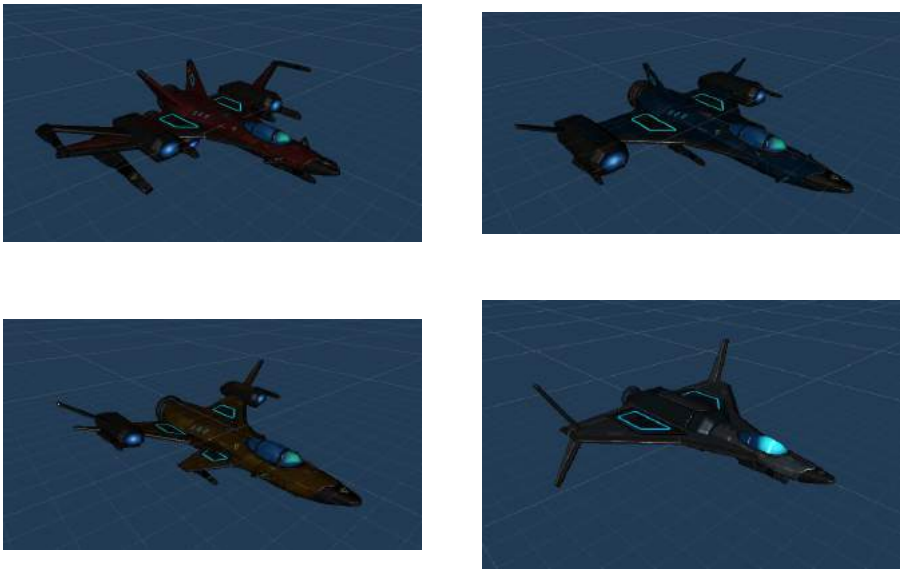
4.3.9 Modèles 3D

4.3.10 Les unités

Year Zero, comme tout bon jeu de stratégie, offre de nombreuses unités. Il faut alors les habiller afin qu'on puisse les différencier. Après de nombreuses recherches de modèles 3D pouvant correspondre à nos besoins, j'ai trouvé un assortiment de ressources qui vont pouvoir constituer une majeure partie des modèles associés aux unités, pour ceux qui joueront les humains tout du moins. C'est ainsi qu'on retrouve pas moins d'une quinzaine de vaisseaux différents, chacun pouvant être décliné en plusieurs teintes, qui à terme viendront marquer les différentes équipes avec la couleur choisie en salle d'attente de partie.

Certains des vaisseaux présents ne feront pas partie du jeu, nous ne les avons pas encore tous finalisés, néanmoins la variété de couleurs et de type de texture différents qui sont à notre disposition nous donnent de nombreuses possibilités pour le jeu en version finale afin de proposer des modèles s'inscrivant dans la charte graphique de Year Zero.
Vous pouvez voir ici des images approchées de certains des vaisseaux de Year Zero.

FIGURE 21 – Vaisseaux de Year Zero en plusieurs couleurs



Un aspect important des modèles de vaisseaux, c'est qu'ils sont modulaires. C'est à dire qu'ils sont décomposable en plusieurs parties constituantes. Ainsi il nous sera plus aisé de matérialiser une destruction voire même une animation de construction de ces vaisseaux.

Outre ces modèles correspondant aux unités même, il y a tout un aspect des unités dans la 3D, qui est souvent oublié, mais nous ne pouvons passer à côté. Il s'agit de l'ensemble des projectiles qui seront tirés par les unités. Cela va du

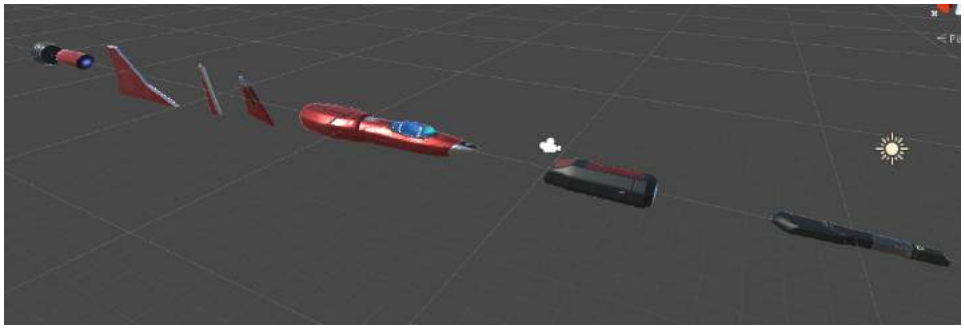
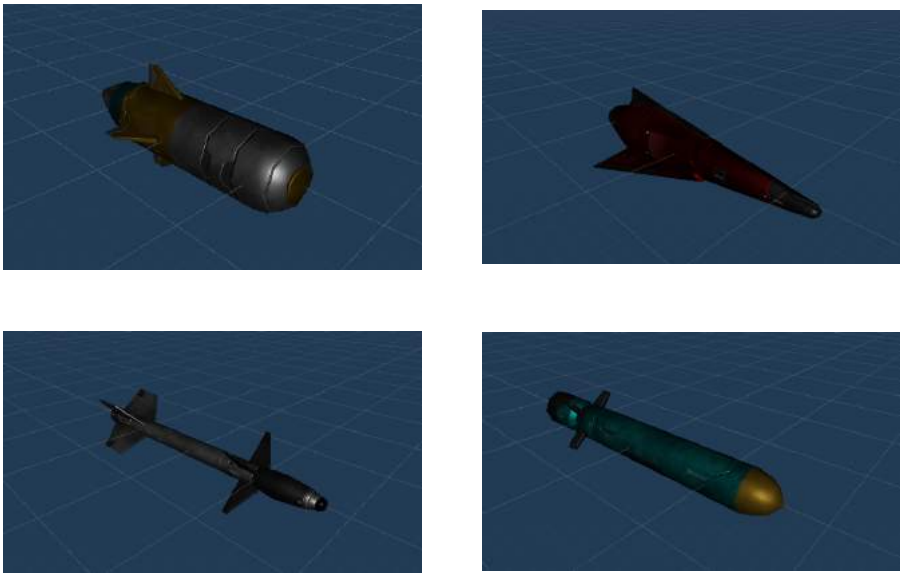


FIGURE 22 – Modularité d'un vaisseau

simple laser aux missiles, en passant par les bombes et autres. Outre le laser qui n'est pas encore dans le jeu, puisqu'il consiste en une animation de particule et que cette dernière n'est pas encore faite, on ne le retrouve pas dans cette version du jeu. De même pour les autres projectiles puisqu'ils seront associés à des effets de particule, des réacteurs pour les missiles, des effets d'explosion etc. Tout ceci devra être présent dans la version du jeu que nous rendrons pour la troisième et dernière soutenance qui marquera la fin du projet. Nous sommes dans l'obligation de fournir un jeu complet pour cet évènement.
Ci-dessous, quelques exemples des modèles de projectiles de tous types qui viendront rejoindre les modèles 3D de Year Zero.



4.3.11 Les bâtiments

Dans Year Zero il n'y a pas moins d'une dizaine de bâtiments différents. La plupart sont du même type, fixes et relativement ressemblant et quelques autres singuliers comme le radar ou encore la tourelle de défense. Donner des modèles à ces bâtiments n'est pas une tâche simple puisqu'ils doivent refléter l'atmosphère du jeu, ne pas briser l'unité d'un environnement spatial que l'on retrouve dans tous les aspects de ce que le jeu nous donne à voir sur l'écran. Ainsi, j'ai ajouté des ressources 3D correspondant à des modules d'un type futuriste sinon spatial qui, mis les uns avec les autres devraient pouvoir nous offrir assez de possibilités pour nos nombreux bâtiments, pour que chacun ait un modèle unique, se distinguant des autres. La plupart ne sont pas encore assignés puisque je suis encore à la recherche de modèles qui pourraient mieux convenir. Une chose à marquer sur la liste des objectifs de la troisième soutenance.

4.4 Modèles additionnels

Un environnement spatial a pour vocation d'être vide, mais il est apparu que remplir ce vide au moins par quelques éléments graphiques additionnels serait une bonne idée et rendrait l'environnement encore plus prenant. C'est pourquoi j'ai rajouté quelques astéroïdes qui vont venir prendre place dans la carte du

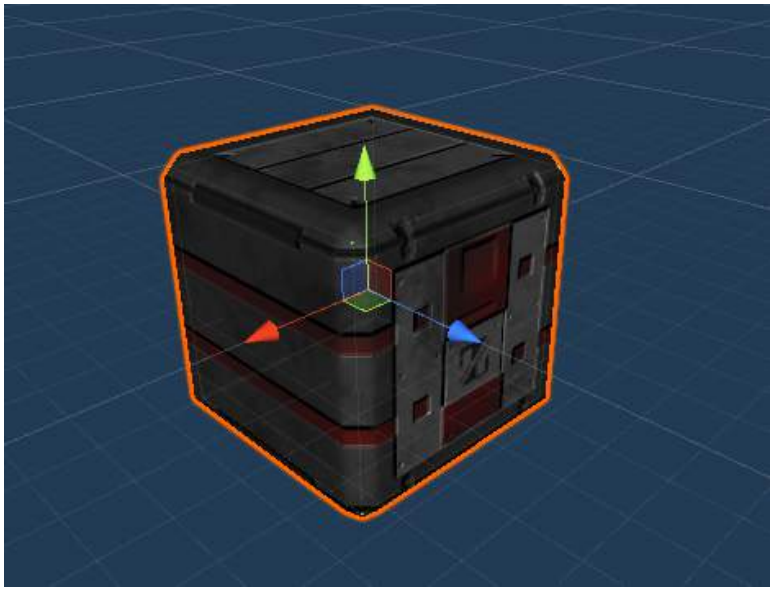


FIGURE 23 – Exemple d'un modèle modulaire

jeu, dans une partie. Ils ne sont que purement décoratifs, on ne peut interagir avec eux, notamment pour récolter des ressources. Ils sont placés plus ou moins haut en fonction du niveau de référence qui est celui des unités et des bâtiments de sorte que l'effet d'environnement spatial soit conservé au maximum et améliorer l'immersion du joueur. Les astéroïdes sont déclinés en plusieurs tailles et possèdent différentes textures que l'on va leur appliquer afin de choisir l'aspect qu'ils ont en fonction du type d'environnement spatial de la partie. Par exemple on pourra appliquer la texture de glace aux astéroïdes lors d'une partie avec comme carte choisie un espace de glace. C'est une des fonctionnalités que je prévois d'ajouter au jeu final, présenté lors de la troisième soutenance.

FIGURE 24 – Petite taille

FIGURE 25 – Taille moyenne

FIGURE 26 – Plusieurs astéroïdes

Toujours dans le soucis de l'amélioration de l'immersion et du rendu 3D du jeu, j'ai aussi ajouté des ressources 3D de débris spatiaux. On y retrouve nombreux débris divers, de bâtiments ou même de vaisseaux. En fonction de la carte on pourra retrouver nombre de ces débris comme élément de décor au même titre que les astéroïdes.
Il est important de noter également que certains de ces débris nous servirons lorsqu'une unité ou un bâtiment sera détruit, nous afficherons alors le modèle 3D de débris correspondant.

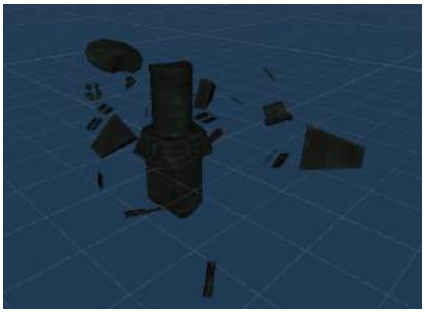


FIGURE 27 – Modèle de débris

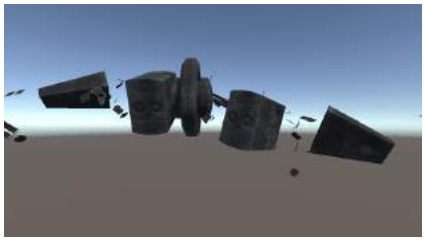


FIGURE 28 – Autres débris

4.4.1 Site internet

J'ai aussi contribué au site internet de Year Zero, particulièrement sur l'ajout de contenu au sein des différentes rubriques présentes.

4.5 Discord Téléchargement

Utilisant régulièrement logiciel Discord pour l'organisation générale du groupe ainsi que pour des réunions vocales avec les membres du groupe, j'ai choisis d'ajouter l'intégration de year Zero au sein de Discord. C'est à dire, que lorsqu'on lance le jeu, Discord affiche sur le profil de la personne jouant au jeu, qu'il y joue. C'est un ajout purement cosmétique et non essentiel mais il contribue à l'image que nous souhaitons donner au jeu, témoignant d'un certain professionnalisme que nous cherchons à atteindre.

4.6 Axel

Je vais ici présenter chaque fonctionnalité que j'ai ajouté dans le jeu, mon ressenti, les problèmes survenus.
Tout d'abord afin de structurer correctement ma partie, je ne vais pas présenter mon travail selon un ordre chronologique mais plutôt par section. Les différents systèmes étant entremêlés les uns aux autres je n'ai pas pu tous les développer les uns à la suite des autres mais en construisant le tout petit à petit. Je vais cependant garder un ordre le plus chronologique possible.

4.6.1 Contrôles

Sélection

Au-delà de créer le projet sur Unity, ma première tâche fut de développer un système de sélection des unités. Le principe est un simple, un clic gauche sur une unité la sélectionne, mais à cela s'ajoute la sélection de plusieurs unités qui survient lorsque qu'on effectue un clic gauche sur la souris et qu'ensuite on la déplace pour former un rectangle permettant de sélectionner l'ensemble des unités dans ce rectangle. J'ai ensuite donné quelques spécificités à cet algorithme, par exemple, si on sélectionne un bâtiment on ne peut en sélectionner qu'un seul, si on effectue un double-clic gauche on sélectionne toutes les unités à l'écran qui sont du même type que celle sous le pointeur de la souris, ou encore, si on dessine un rectangle autour d'unités qui ne nous appartiennent pas on n'en sélectionnera finalement qu'une seule. Aussi, un clic droit sur une unité va exécuter l'interaction adéquat. Cet algorithme est un des plus complexe qu'il m'ait été donné de développer dans le cadre de ce projet tant il prend en compte de paramètres, j'ai dû le penser et le repenser continuellement au fur et à mesure que je lui ajoutais des fonctionnalités.
Ma première difficulté fut de savoir comment je pourrais sélectionner l'unité sous le pointeur de la souris. Après quelques recherches, j'ai découvert le principe du Raycast, cette fonctionnalité d'Unity permet d'envoyer un rayon depuis un point et suivant un vecteur directeur défini et renvoie l'objet rencontré par le rayon. J'envoie donc un rayon depuis la position de la souris et dirigé vers le point juste en dessous de la souris pour accéder à l'unité et ainsi la sélectionner.
Le système de multi-sélection est plus simple mais aussi beaucoup plus gourmand, il va parcourir la liste des unités et transposé leur coordonné 3D en 2D puis les sélectionner si elles sont dans le rectangle dessiné.

Player control system

Au fur et à mesure que j'ajoutais des fonctionnalités au jeu, il m'est apparu qu'il fallait faire un système pour gérer différents profils de contrôles, par exemple lorsqu'on veut placer un marqueur sur la carte on appuie sur le bouton associé et on passe alors dans un nouveau profile de touche très simplifier dans lequel un clic gauche sur la minicarte va poser le marqueur et un clic en dehors de la minicarte ou bien un appuie sur la touche ECHAP va annuler le mode marqueur pour repasser aux contrôles classiques. Sans ce système il serait compliqué d'isoler certaines fonctionnalités et par exemple ouvrir le menu pause n'empêcherait pas de sélectionner des unités ou de déplacer la caméra ce qui pose un problème.

Formation System

Cet algorithme ne faisait pas partie de nos priorités mais son utilité est indiscutable tant d'un point de vue technique que pour le joueur. En effet il s'agit là d'un système permettant de déplacer ses unités en formation rectangulaire. Ce système évite que toutes les unités essayent de rejoindre le même point lorsqu'on les fait se déplacer car techniquement une seule va l'atteindre avant les autres et va ainsi empêcher par se présence les autres de rejoindre ce point menant ces unités à toutes se bousculer à l'infini pour atteindre leur destination. L'algorithme est assez simple et défini pour chaque unité une destination différente. Le seul problème auquel j'ai dû faire face était le cas où on ne sélectionnait pas un multiple de 5 unités, en effet les rangés étant de 5, la formation était relativement laide et surtout si on sélectionnait une seule unité elle n'allait pas au point désiré mais quelque centimètre au-dessus car elle commençait la formation. J'ai donc modifié l'algorithme de sorte qu'au lieu de former une ligne dans l'ordre 1 2 3 4 5, il la forme dans l'ordre 5 3 1 2 4.

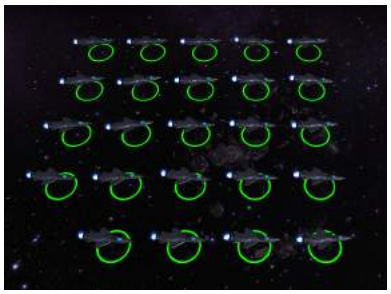


FIGURE 29 – Formation

Camera controller

Il est un des premiers algorithmes que j'ai implémentés car il est essentiel et simple. Par défaut la caméra est orientée à 60 ˚vers le bas pour avoir une vue assez générale de la partie mais offrant tout de même une certaine profondeur. La suite est simple, si on appuie sur les classiques touches Z Q S D ou bien que l'on place la souris sur les bords de l'écran on va pouvoir déplacer la caméra. Une autre fonctionnalité implémentée plus tardivement permet grâce la molette de la souris de zoomer en quelque sorte la caméra, en effet celle si va se rapprocher du sol et réduire son angle rotation de sorte à se retrouver quasiment à l'horizontale. Cette vue est plus esthétique qu'autre chose car elle n'est pas très pratique mais offre une vue plus cinématographique et épique sur la partie.

4.6.2 Managers

J'ai nommé manager toutes les classes comportant un singleton. Ce singleton permet de limiter le nombre d'instance de cette classe à une seule et de la rendre accessible par absolument tous les autres scripts simplement. Ces algorithmes sont en général primordiaux et demande un accès récurrent de la part d'autres script.

Chat Manager

Ce programme implémenté assez tardivement car non essentiel permet de gérer de manière générale tous les messages envoyés par les joueurs. La partie visuelle de cet algorithme sera détaillé plus loin dans le rapport. Concrètement, à chaque fois qu'un joueur envoie un message il sera stocké ici grâce à 2 informations, le joueur qui l'envoie et son contenu. Puis il sera ensuite envoyé à tous les autres joueurs de la partie grâce à un RPC (voir partie sur le multijoueur).

Instance Manager

Celui-ci permet de faire le lien entre diverses actions locales et des informations multijoueur. En début de partie, il va récupérer les informations du joueurs actuelle grâce à ses custom properties (voir multijoueur) définies dans la salle d'attentes (équipe, race, couleur, coordonnés de départ). Puis il va instancier les troupes de départ aux bons coordonnés. Ce manager gérera ensuite la notion d'échec qui surviendra lorsqu'un joueur n'a plus d'unité ou bien le cas où un joueur se déconnecte entrainant la déconnexion de tous les autres.

Player Manager

Ce manager va gérer tous ce qui attrait aux ressources, c'est en effet ici qu'on ajoute ou retire de la ressource et qu'on trouve les fonctions permettant de payer des constructions ou de la production d'unités. Il va aussi gérer les différentes stations spatiales de sorte que les unités comme les Constructeurs puissent ramener leurs ressources à la station spatiale la plus proche.

4.6.3 Jeu

Gameresource

Une Gameresource est tout simplement une ressource, les minerais, l'énergie et la nourriture en sont. Elle contient simplement son nom, sa quantité et diverses méthodes pour modifier ses valeurs.

Population

Cette GameResource particulière gère la population. Elle est constituée d'une population maximum actuelle et d'une population actuelle. Si la population actuelle est égale à la population maximum actuelle alors on ne peut plus créer d'unité mobile. Pour augmenter ce maximum il faut construire des maisons.

Placement
Il s'agit là du système de placement de bâtiments. Il est décomposé en 3 sous systèmes :

Raycaster :
Un Raycaster va envoyer un Raycast vers le bas et dire s'il rencontre ou non le sol, si non ou bien si le sol est soit trop proche soit trop loin alors une variable booléenne va signaler que la case sur laquelle se trouve le Raycaster n'est pas disponible pour la construction.

DetectionCell :
Celle-ci est composé de 5 Raycaster disposé comme les points du cinq sur un dé de sorte à pouvoir tester la case assez précisément. Une DetectionCell représente une case de la Carte et va observer chacun de ses Raycaster, si au moins un seul d'entre eux se présente comme non disponible à la construction, la case au départ de couleur verte va passer rouge.

PlacementGrid :
Une PlacementGrid va être généré lorsqu'on veut poser un bâtiment, elle va alors obtenir la taille du bâtiment en case et va générer autant de DetectionCell que le bâtiment prend de case. Si au moins une seule des DetectionCell est au rouge alors la PlacementGrid empêchera le joueur de poser son bâtiment. Le centre de la PlacementGrid correspond au pointeur de la souris mais avec le système de case.

Ce système a demandé une certaine réflexion avant d'être développé, j'aurais pu me contenter d'un algorithme empêchant simplement de poser deux bâtiments l'un dans l'autre mais j'avais plus d'ambition et voulais un système qui montre au joueur quelle(s) case(s) pose(nt) problèmes. Étant dans l'espace la distance entre le sol et le Raycaster est inutile mais le système est là et bien fonctionnel. D'autre part le système de case est un artifice. Il n'y a pas de case à proprement parler mais simplement un calcul qui va créer un modulo des coordonnés sur lesquels on peut placer un bâtiment. Par exemple si le modulo est de 5, et que le joueur place son pointeur en $x = 12$ alors au lieu que le centre de la PlacementGrid soit en $x = 12$ il sera en $x = 10$, si le pointeur est en $x = 13$ celui de la PlacementGrid sera en $x = 15$ et ainsi de suite.

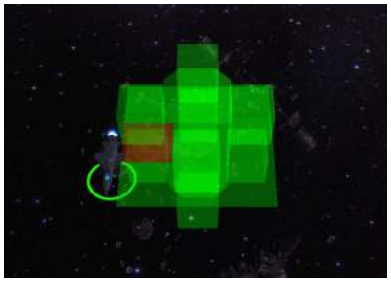


FIGURE 30 – Système de construction

Tasks
Le système de Tasks a été développé spécialement pour les bâtiments afin de leur donner la possibilité d'exécuter des tâches comme son nom l'indique. Au départ je voulais créer plusieurs types de tâches, celles de production et celles d'amélioration. En fin de compte seule la première a été implémenté, l'autre étant devenu l'arbre de compétences. Une tâche de production permet simplement de produire une unité. Chaque tâche se paye en ressource et prend un certain temps pour s'effectuer.
Bien que simple à expliquer, cet algorithme m'a pris pas mal de temps car il mêle beaucoup de système précédemment développer, il a fallu corréler le tout et évitant de créer des bugs. Le tout en mettant bien à jour l'interface afin de lui donner les bonnes informations.

4.6.4 menus

Main Menu et In-Game Menu

Une fois le cœur du jeu bien amorcé j'ai voulu incorporer le multijoueur rapidement, mais il me fallait d'abord créer des menus pour cela. Étant donné leur nombre conséquent j'ai dû créer un algorithme très général me permettant de simplifier au maximum l'implémentation de nouveau menu. En effet, au-delà de faire les choses correctement pour le joueur, il faut aussi créer des outils assez puissant et intelligent pour faciliter le travail du développeur.

Menu principal :
Permet de lancer le mode solo, multijoueur, d'aller dans les options, de voir les crédits, ou encore de quitter le jeu. Menu solo :

Permet de choisir entre le mode campagne, le mode partie rapide et le mode sans fin.
Menu Campagne :
Permet de revoir la cinématique d'introduction, de faire le tutoriel ou bien de lancer une des missions de campagne du jeu. Menu multijoueur :
Permet de choisir entre rejoindre une partie ou en créer une. Pour rejoindre une partie il suffit de taper son nom et de cliquer sur Join.

Menu de création de partie :
Permet de créer une partie en choisissant la carte, et si le joueur créer une partie multijoueur de donner un nom à la partie ainsi que définir le nombre maximum de joueur (IA comprise).

Menu Options :
Permet de régler certains paramètres. Il est trié en plusieurs catégories :
-Gameplay : Permet de définir la vitesse de défilement de la caméra avec le clavier et la souris, de désactiver le défilement à la souris et d'activer ou non les bulles d'aides. -Pseudo : Permet de choisir son pseudo
-Vidéo : Permet de choisir la résolution, d'activer ou non le plein écran, et de choisir le niveau de qualité graphique du jeu.
-Son : Permet d'ajuster les différents volumes sonores du jeu

J'ai reproduit le même menu durant la partie, ils manquent seulement l'option de changement de pseudo parce qu'il est logique qu'il soit impossible de le changer durant une partie.

AlliesMenu
Le menu des alliés permet d'envoyer des ressources à ses alliés. Il utilise le système de RPC (voir multijoueur).

loading times
Les temps de chargement ont été ajouté et demeurent relativement peu utile. En effet, le jeu n'étant pas très gourmand en ressource les textures n'étant pas dans de très hautes résolutions et les modèles ayant assez peu de polygone, les temps de chargement sont à peine perceptibles, mais le système est bien là et opérationnel. Il utilise la faculté d'Unity à pouvoir charger une scène tout en maintenant l'actuelle utilisable, et pour ce qui est de la barre de progression du chargement, Unity nous fournit une variable entre 0 et 1 donnant cette progression.

4.6.5 Interface

Dans cette sous-partie je vais expliquer tout ce qui attrait à l'interface durant la partie.

Cards

J'ai nommé Carte les petites fiches de chaque personnage sélectionné. Le panneau qui les contient permet alors d'affiche pour chaque unité une image de son modèle ainsi que sa jauge de vie pour avoir une vue d'ensemble sur les troupes sélectionnées. Son utilité n'est pas seulement visuelle puisque cliquer sur une de ces cartes permet de sous-sélectionner une unité. Cela signifie que les actions disponibles s'effectueront sur l'unité sous-sélectionné et non l'ensemble des troupes.

FIGURE 31 – Cards

Chat

La partie visuelle du chat est découpé en 2 panneaux : Le premier ne propose aucune interaction, il est situé sur la gauche de l'écran de jeu et affiche les messages envoyés mais ne les gardes qu'un temps, au bout de quelques secondes les messages disparaissent. Le second est accessible depuis un menu et affiche quant à lui tous les messages envoyés sans jamais les supprimer. Il offre une zone de texte pour pouvoir envoyer des messages.
L'autre méthode pour envoyer des messages est d'appuyer sur entrée dans le jeu ce qui fait apparaître une zone de texte pour écrire son message.
Ces 2 panneaux utilisent le Chat Manager qui sert de banque de message en quelque sortes.

advancement bar

Il s'agit simplement d'une barre de progression qui survient lorsqu'on sélectionne

un bâtiment en construction et qui donne l'avancement de celle-ci.

description panel
Apparait lorsqu'on passe la souris sur un bouton pour construire un bâtiment ou instancier une unité. Il contient le nom de l'objet à créer, son coût en ressource ainsi qu'une petite description de l'objet.

FIGURE 32 – Description Panel

Help
Il s'agit d'un petit panneau qui va afficher l'utilité de certains boutons lorsqu'on passe le pointeur de la souris dessus car certains boutons utilisent des icônes qui peuvent prêter à confusion. Ce panneau est désactivable dans les options. Il survient notamment lorsqu'on passe la souris sur les icônes de la minicarte.

FIGURE 33 – Bulle d'aide

Help Menu

Ce menu contient de multiples astuces pour permettre au joueur d'avancer correctement dans la partie. Il peut aider au cas où le joueur oublierai d'utiliser certaines mécanique de jeu par exemple.

jobless button

Ce bouton sert à indiquer si oui ou non certains constructeurs son inoccupés, et si oui alors un clic sur ce bouton sélectionne cette unité et déplace la caméra au-dessus de celle-ci.

Monodescr

Ce panneau survient lorsqu'on sélectionne une unité seulement et n'affiche que son nom ainsi que la quantité de ressource qu'il transporte s'il s'agit d'un constructeur.

Portrait

Le portrait affiche la vie de l'unité sous-sélectionné ainsi que son modèle. Il ne s'agit pas là de l'affichage d'un modèle issue des Assets du jeu mais d'une caméra qui va se place devant l'unité sous-sélectionnée, le portrait affiche tout simplement ce que voit cette caméra.

FIGURE 34 – Portrait

Resources

Le panneau de ressource va générer au lancement du jeu autant de panneau affichant le nom et la quantité de ressources que de ressources ajoutées au jeu.

Tools

Le panneau des outils et une des fonctionnalités les plus importantes du jeu. Un outil (tools) désigne ici un bouton associé à une unité et qui va effectuer une action avec cette unité. Par exemple : Celui pour déplacer l'unité, pour l'arrêter, pour afficher le menu de construction, pour instancier une unité, pour aller réparer un bâtiment et ainsi de suite.
Ce panneau allant servir très souvent il se devait d'être bien codé, pour être assez puissant mais aussi facile à utiliser pour le développeur. Dans les faits, pour certains outils particuliers il faut créer des scripts particuliers mais pour les unités il suffit de glisser déposer le préfabriqué de cette unité dans la liste de tools d'une autre unité pour qu'elle puisse générer un tools qui créer une Task d'instanciation par exemple.

FIGURE 35 – Outils (Tools)

floating life bar
Pour chaque unité instanciée, une floating life bar va être généré et va faire apparaitre la vie d'une unité lorsqu'on passe la souris dessus.

minimap La minimap (ou mini-carte) est en réalité une simple caméra placée au-dessus du jeu et qui en donne une vue d'ensemble. Chaque unité se voit attribué une icone qui ne sera affiché que sur la caméra. Elle contient par ailleurs d'autres fonctionnalités :
-Si une unité est dans le brouillard de guerre elle ne s'affichera pas sur la minimap
-Un clic gauche sur la minimap déplace la caméra de sorte qu'elle regarde l'endroit où le pointeur est
-Un clic droit déplace les unités sélectionnées à l'endroit pointé sur la minimap
-Un rectangle indique la zone observée par la caméra
-Un bouton permet de placer un marqueur sur la minimap, tous les joueurs de

FIGURE 36 – Floating Life Bar

l'équipe le verrons aussi
-Un bouton permet d'afficher ou non le fond de la minimap
-Un bouton permet d'afficher ou non les unités
-Un bouton permet d'activer ou non le système de mise en formation des troupes
-Un bouton permet de changer la gestion de l'affichage des couleurs selon l'équipe.

FIGURE 37 – Mini-Carte

Selectionbox
Il s'agit du rectangle qui va montrer au joueur quel zone il est en train de sélectionner.

TemporaryMessage
Placée en haut droite de l'écran, cette zone va afficher des messages temporairement pour justifier au joueur pourquoi il ne peut pas encore effectuer une

action. Par exemple s'il essaye de construire un bâtiment alors qu'il n'a pas assez de ressource.

4.6.6 Units

Dans cette sous-partie je vais expliquer tout ce qui attrait aux unités de manière générale, les unités plus particulières comme les bâtiments ou les unités mobile seront détaillé plus loin dans le rapport.

Interactable
Il s'agit de l'unité de base, elle ne possède aucune méthode ni attributs, elle définit seulement que l'on peut interagir avec cet objet en faisant un clic droit dessus.

Selectable
Elle hérite d'Interactable, et permet de rendre une unité sélectionnable. En effet cette classe définit la notion de Highlight (quand on passe la souris sur une unité sans la sélectionner, un cercle de couleur va s'activer pour montrer que la sélection est possible) et de sélection (le cercle est opaque). Ce cercle va avoir une couleur différente selon que l'unité appartient ou non au joueur, à son équipe ou aux équipes ennemies. La classe contient aussi le nom de l'objet ainsi que son coût, sa description et le temps requis pour le produire.
Cette classe permet aussi de générer le champ de vision, c'est-à-dire jusqu'à quel distance le brouillard de guerre (voir brouillard de guerre) est dissipé autour de cette unité.

FIGURE 38 – Normal

FIGURE 39 – Highlighted

FIGURE 40 – Selected

DestructibleUnit
Elle hérite de Selectable, et permet de rendre une unité destructible. En effet cette classe définit la notion de jauge de vie et de mort. C'est ici qu'on définit

la vie d'une unité, qu'on peut la soigner et la détruire.

Resource Unit
Elle hérite d'Interactable. Cette unité contient une certaine quantité d'un certain type de ressource et se détruit lorsqu'elle est vide.

4.6.7 Buildings

Constructedunit
Il hérite de DestructibleUnit, c'est le bâtiment de base. Il possède un Task System (voir Tasks), et il se voit attacher un InConstructionUnit (voir InConstructionUnit) ainsi qu'un modèle du bâtiment mais tous vert et transparent (appelé Ghost, il sert à afficher le bâtiment lorsqu'on souhaite le placer). Un autre système lui est attaché, celui de la réparation, en effet si le bâtiment n'a pas son maximum de vie, alors on peut envoyer un ou plusieurs constructeurs le réparer.

InConstructionUnit
Il hérite aussi de DestructibleUnit, chaque ConstructedUnit a un InConstructionUnit, il s'agit du même bâtiment excepté qu'il est la version en construction, cela permet de gérer des scripts aux fonctionnements totalement différents de manière indépendante et donc plus simplement.
Il possède à peu près le même système de réparation que le ConstructedUnit mais pour la construction du bâtiment en lui-même.

Production unit
Ce bâtiment va produire des unités, sa spécificité et des posséder une bannière de ralliement, c'est-à-dire que le joueur va placer une bannière attachée à ce bâtiment et les unités produite vont automatiquement essayer de rejoindre la bannière une fois instancié.

House
Ce bâtiment augmente la population maximale.

Radar
Ce bâtiment est constructible un nombre limité de fois (ce nombre peut être augmenté dans l'arbre des compétences). Il ne fait rien de particulier si ce n'est prévenir le joueur quand une unité ennemie entre dans son champ de vision. Un champ de vision volontairement plus grand que les autres bâtiments.

Station Spatiale

C'est le bâtiment principal de la base, il est l'endroit où les mineurs vont déposer leurs ressources. C'est lui qui permet de produire des constructeurs.

EnergyFarm

Ce bâtiment génère continuellement de l'Énergie.

Farm

Ce bâtiment produit de la nourriture qui doit ensuite être récupéré par les constructeurs et amené à la station spatiale.

Asteroid

Cette unité hérite de ResourceUnit, elle contient un nombre définit de minerais (une des ressources du jeu) et s'autodétruit lorsque tout a été miné.

Laboratoire

Ce bâtiment génère continuellement des points de technologie (pour l'arbre de compétence).

Movable unit

Cette classe hérite de DestructibleUnit et contrairement au bâtiment elle est caractérisée par la possibilité de se déplacer. Elle peut aussi se faire hacker (L'unité appartient désormais au joueur qui hack).
Chaque unité mobile est équipée du système de patrouille, grâce à un Tools, on peut dire à une unité mobile de faire des allés et retours continuels entre son point actuel et le lieu pointé par la souris.

Constructeur

Cette unité mobile a beaucoup de fonctionnalités :
-Récolte : Elle fait des allés et retours entre son lieu de récolte et la station spatiale la plus proche en ramenant des ressources à chaque fois
-Construction et Réparation : Elle peut construire un bâtiment ou bien le réparer (plus il y a de constructeur qui construisent/réparent un bâtiment et plus cela va vite)
Cette unité m'a demandé beaucoup de temps et de réflexion pour bien m'organiser et que les divers comportements associés n'influent pas les uns sur les autres au risque de créer des bugs. (Par exemple, si un constructeur se voyait ordonner la construction d'un bâtiment alors qu'il en réparait un autre, il ne fallait pas que les 2 comportements se mélange sinon l'unité aurait eu un com-

portement chaotique). Chaque module est simple mais gérer tous ces modules est plus complexe.

Mobile Medical Station
Cette unité soigne les unités mobile alentours.

Hacker
Cette unité peut par définition en hacker d'autre (cela les fait changer de camps), mais aussi lancer des bombes électromagnétique ainsi que dévoiler une zone pendant un certain temps.

Basic troop
Troupe de combat par défaut son attaque et sa défense sont équilibrés.

Light Troop
Troupe de combat, son attaque est puissante et sa défense faible mais il est très rapide et consomme très peu de population.

Bombardier
Unité de combat lente mais robuste et qui envoie des missiles très puissant mais à une cadence faible.

Croiseur
Unité unique très lente, très résistante, mais qui améliore les unités alliées alentours et une fois la bonne compétence débloquée sert de relais pour les bâtiments d'instanciation de vaisseaux. C'est-à-dire que les vaisseaux crées apparaîtront au niveau du croiseur.

4.6.8 Graphics

Brouillard de Guerre
Le principe du brouillard de guerre est simple, la carte du jeu est recouverte d'un brouillard, chaque unité à un champs de vision sous forme d'un cercle plus ou moins grand et qui va dissiper ce brouillard. Ce système est en réalité composé de 2 systèmes l'un est purement graphique et l'autre uniquement orienté programmation.
Graphiquement il s'agit d'une couche qui assombrit le jeu et grâce au système de shader intégré à Unity, on peut simuler qu'une zone est éclairé. Techniquement, lorsqu'une unité ennemie n'est dans le champ de vision d'aucune unité alliée,

alors ses textures, son modèle, son icône de minimap sont désactivée, l'unité est invisible.
Ce système m'a posé beaucoup de problème car il utilise les shaders que je connais très peu et qui ne se codent pas en C# mais Cg HLSL que je ne connais pas. De plus très peu de forum et de tutoriels existent à propos des brouillards de guerre. Il a fallu du temps et des recherches pour toutes la parties visuelle, la partie code étant plus traditionnelle.

FIGURE 41 – Fog Of War

4.6.9 Waiting room

Cette partie va détailler le fonctionnement de la Waiting Room, c'est-à-dire la salle d'attente dans laquelle les joueurs attendent que d'autres les rejoignent ou que le créateur de la partie la lance. Cette salle va gérer indépendamment chaque joueur à partir du moment où il entre dans la salle d'attente. Tout d'abord un panneau de paramètre visible pour tous les joueurs va s'afficher, seul le joueur qui en est le propriétaire peut le modifier, ce panneau va permettre au joueur de définir, sa classe, son équipe, sa race et sa couleur. Le créateur de la partie peut ajouter des Bots (Intelligence artificielle) et lancer la partie une fois que tous les joueurs ont coché la case ready. Une fois cette case coché, un joueur ne peut plus modifier ses propriétés. Au lancement de la partie, le Players Manager va répartir les coordonnées de départ de chaque joueur de sorte que les joueurs étant dans la même équipe se retrouve proche les uns des autres.

4.6.10 Arbre des Compétences

Au-delà de la construction nous avons voulu qu'une notion de progression soit présente dans le jeu et qui plus est qui permette aux joueurs de se différencier de sorte que le vainqueur ne soit pas seulement déterminé par la stratégie

FIGURE 42 – Salle D'attente

en combat mais aussi par tout le déroulement de la partie. Nous avons donc opté pour un arbre de compétence. Celui-ci est découpé en plusieurs sous-arbre distincts dédiés chacun à un secteur particulier que sont :
- Attaque légère
- Attaque lourde (bombardier)
- Recherche et développement (Hacker)
- Croiseur
- Économie (Construction, ressources)
- Défense (tourelle)

Chacun de ces arbres permet donc de débloquer des unités ou des compétences mais aussi d'améliorer certaines statistiques comme la vitesse de construction ou les dégâts par seconde d'une unité. De plus ces arbres sont restrictifs, c'est-à-dire que lorsque plusieurs compétences se trouve à la même hauteur de l'arbre, choisir l'une d'entre-elle va bloquer toutes les autres. Le but étant qu'en fin de partie, il n'y ai que très peu de chance que les joueurs aient fais les mêmes choix et se retrouve finalement avec la même base et la même armée. L'achat de compétence coûtent des points de technologies.

4.6.11 Online

Photon

Pour gérer le multijoueur dans notre jeu nous avons choisi d'utiliser l'asset Photon PUN 2 notamment pour son accessibilité. J'ai suivi les conseils des InfoSpé

FIGURE 43 – Arbre des Compétences

en implémentant très rapidement le multijoueur. PUN offre plusieurs méthodes pour synchroniser en ligne les divers éléments du jeu, tout d'abord on peut choisir d'envoyer un stream qui contient diverse variable à synchroniser, sinon on peut utiliser les RPC qui permettent d'exécuter une fonction sur la même instance d'un objet chez les autres joueurs.

4.6.12 Préparation à l'IA

Avant de commencer le cœur de l'IA, j'ai d'abord dû préparer le jeu à son arrivée. N'ayant jamais développé d'IA pour ce type de jeu mes algorithmes n'étaient pas prévu pour cela. En effet jusque-alors l'identification du joueur auquel appartient une unité passait par le système implémenté pour le multi-joueur, or il n'est pas possible de créer des joueurs « factices » correspondants aux IA. L'hôte de la partie (celui qui l'a créé), doit ainsi contenir toutes les IA localement. J'ai dû alors implémenter des méthodes pour identifier si une unité appartient à un joueur ou une IA et réagir en fonction.
Ensuite, j'ai dû implémenter des outils pour pouvoir permettre à l'IA de contrôler son jeu. En effet, des fonctionnalités comme l'instanciation d'unité ou la construction de bâtiment ont été conçu à la base pour le joueur qui est équipé d'une souris et d'un clavier notamment. J'ai alors développé différents module pour chaque type d'action :

-L'Armée : Cet objet va permettre de savoir si l'armée de l'IA est prête pour l'attaque ou non, et ainsi l'envoyer attaquer.
-Les Constructeurs : Ce module permet de récupérer des constructeurs en fonction de leurs actions en cours (ne fait rien, mine ou construit) ainsi que les répartir équitablement entre toutes les tâches possibles.
-La Construction : Permet à l'IA de lancer simplement l'ordre de construire un bâtiment. La position du bâtiment généré en forme d'escargot. C'est-à-dire que les bâtiments sont construits autour de la Station Spatiale initiale pour former un carré qui deviendra toujours plus grand au fur et à mesure de la construction de bâtiment. Cependant, l'IA vérifie avant de construire que le terrain est apte. Ainsi, si un minerai ou un bâtiment est déjà présent elle passera automatiquement à la position suivante.
- L'Instanciation : Tandis que le joueur doit sélectionner un bâtiment pour lui demander d'instancier une unité, l'IA, elle, va demander une unité, et ensuite, elle déduira dans quel bâtiment il est idéal d'initialiser la tâche.
-Le Minage : Permet d'envoyer un constructeur miner. L'algorithme déterminera alors l'astéroïde ou la ferme optimale sur laquelle envoyer la troupe. Pour la déterminer, l'algorithme va attribuer un score à chaque bâtiment de ressource compatible et prendre celui qui à le score le moins élevé. Ce score est établi grâce à la distance entre la Station spatiale et le bâtiment de ressource ainsi que par le nombre de constructeurs déjà en train de miner dessus.

Le but de ces outils est de permettre à l'IA de contrôler sa partie, on peut imager ces outils comme le corps du Bot, et l'IA comme étant sa tête. Une fois ces outils terminés j'ai pu passer au développement de l'IA c'est à dire, les utiliser intelligemment.

4.6.13 IA

Avant de parler du fonctionnement de l'IA je vais m'attarder sur les réflexions et le chemin qui m'a mené à cette vision de l'intelligence artificielle.

Histoire :

N'ayant aucune expérience en intelligence artificielle, et face à la difficulté qu'est de développer une IA pour un jeu de stratégie nous avons voulu en faire une assez modeste en termes de faculté de raisonnement. En effet dans un jeu de stratégie l'IA doit savoir gérer beaucoup de chose, elle doit anéantir les équipes adverses mais pour cela elle doit d'abord développer sa base, gérer ses ressources, créer assez de constructeurs, découvrir de nouvelles ressources et

construire tel ou tel bâtiment pour arriver à ses fins. Ainsi j'ai privilégié de faire une IA simple mais fonctionnelle au détriment d'une IA plus ambitieuse au risque d'échouer.
Ma première idée fut de mélanger scripts et réactions, en effet je n'avais aucune idée de la manière pour faire en sorte que l'IA gère bien ses ressources et fasse les bons choix. Concrètement cela devait se traduire par une suite d'action que l'IA allait devoir effectuer de manière scriptée avec quelques petites libertés. Les grandes lignes de son comportement devaient être scriptées, notamment la construction de tel ou tel bâtiment, tandis que d'autres actions comme la création de nouveau constructeurs devait être automatiquement géré par l'IA. Le premier défaut de cette méthode est de faire durer cette IA dans le temps, tant qu'elle n'a pas gagné elle doit continuer d'améliorer sa base ainsi que de former des armées, mais si l'IA suit des actions scriptées elle se serait arrêtée tôt ou tard. Pour remédier à cela j'ai décidé que la suite d'action que devait effectuer l'IA serait très générique et qu'une fois que toutes les actions auraient été effectuées, l'IA allait recommencer à suivre les même actions (à quelques exceptions près). Cependant un défaut majeur persistait, il nous aurait fallu créer cette liste d'action. Cela est possible mais très compliqué, elle se devait d'être très bien pensée pour ne pas brider l'IA. Pour pallier cette difficulté, j'ai voulu donner plus d'autonomie à l'IA, lorsqu'elle aurait manqué de ressources, qu'elle détermine si elle a besoin de créer de nouvelle unité ou bien d'attendre ou bien de mieux les répartir, et lorsque ce cas se serait présenté, de reporter la tâche actuelle à la fin de la liste. Pour pousser ce principe encore plus loin j'ai voulu faire en sorte que, si l'IA voulait créer une unité, elle devait avoir construit le bâtiment adéquat. Cela est aisément automatisable et c'est alors que j'ai compris où cette automatisation allait me mener. En réalité il est assez simple pour l'IA de déduire d'elle-même ce dont elle a besoin, c'est ainsi que j'ai changé ma manière de voir les choses pour arriver à l'IA que je m'apprête à présenter.

IA :

Voici le fonctionnement de l'IA : en début de partie, un objectif lui est donné, celui de former une armée prédéfinie. L'IA va alors utiliser tous les moyens en sa possession pour accomplir cet objectif ; Si elle ne possède pas les bâtiments nécessaires elle les construira, mais pour cela elle a besoin d'un constructeur qu'elle créera alors. Si elle manque de ressources elle pourra ou non demander de nouveau constructeurs, ou bien mieux les répartir, ou encore attendre.
Le fonctionnement est simple, chaque tâche que l'IA doit effectuer est stocké dans une pile. Lorsque l'IA accomplie une tâche elle dépile pour effectuer celle

qui sort de la pile. Si elle ne peut pas encore l'effectuer pour tel ou telle raison, elle va empiler de nouveau la tâche puis en créer une nouvelle qui répondra aux besoins de la tâche initiale.
Chacune de ces tâches se met en quelque sorte en erreur lorsqu'elles ne sont pas effectuables, cela est détecté par l'IA et, en fonction de l'erreur, une tâches adéquate sera créée. Une fois que l'IA aura terminé son armée, elle va détecter quelle est la base ennemie la plus proche et l'attaquer. Une fois l'assaut lancé, seule la mort des troupes l'arrêtera. En effet, si une armée détruit une base adverse elle ne rentrera pas mais attaquera la suivante jusqu'à la victoire ou la mort. Pendant ce temps, dans la base, l'IA est déjà en train de préparer sa nouvelle armée au fur et à mesure que la précédente voit son nombre de troupes diminuer.

Les seuls éléments prédéfinis sont :
- Les armées : Nous définissons nous-même ce que l'IA devra créer comme troupes. Lorsqu'une armée est terminée, l'IA passe à la suivante, et une fois que la dernière a été créée, l'IA créera à nouveau cette même armée jusqu'à la fin de la partie. - Le nombres de bâtiment maximum : En effet, pour éviter que l'IA ne construise le maximum de maison en début de partie, ou bien qu'elle construise une nouvelle station de combat à chaque fois que les autres sont pleines, j'ai défini un nombre maximum pour chaque bâtiment, et ce nombre va augmenter à chaque que l'IA termine une armée. Cela permet de faire en sorte que la base de l'IA grossisse continuellement et régulièrement.

4.6.14 Combat

Pour cette soutenance je me suis aussi occupé du système de combat. Le principe est simple, mais deux difficultés majeures sont à prendre en compte, le comportement ainsi que le multijoueur.
Etant donné que les troupes sont des vaisseaux, il est normal que ceux-ci se défendent en se tirant dessus avec des projectiles. Chaque unité de combat observe dans un certain rayon autour d'elle, si une unité ennemie entre dans son périmètre elle l'attaquera si elle ne fait rien d'autre. Le comportement des unités est délicat lorsqu'une ennemie approche, doit-elle l'attaquer ? continuer ce qu'elle fait ? fuir ? Elle se contentera de riposter si elle ne fait rien et sinon continuera ce qu'elle fait.
Lorsqu'une unité en attaque une autre elle va instancier à intervalle régulier des projectiles vers l'ennemi qui lui feront des dégâts s'il le touche. Les troupes restent toujours face à leur ennemi et en cas de fuite le suivent tant que la distance entre les deux est suffisamment petite. La fréquence de tir, ainsi que

les dégâts par projectile varient selon les unités, et selon l'arbre de compétence. Une de mes difficultés a été de gérer cela en multijoueur, tout d'abord il n'est pas très optimisé d'instancier les projectiles de chaque troupe en multijoueur et de suivre leurs positions, notamment lors de grandes batailles. Ainsi, étant donné que les balles ne se déplacent que dans une direction, et qu'elles n'en changent jamais, j'ai fait en sorte que lorsqu'une balle est instanciée chez un joueur elle le soit chez les autres mais que le joueur propriétaire de la balle communique aussi les force et la direction de la balle aux autres joueurs qui la feront alors se déplacer localement chez chacun. Ceci n'a qu'un but purement visuel puisque c'est le joueur propriétaire qui détruira ou non la balle pour tous les autres lorsque celle-ci touche un joueur ennemi. Le désavantage de cette méthode à l'inverse de celle qui consiste à synchroniser les positions en ligne est que chez les autres membres de la partie cette balle peut ne toucher aucune unité visuellement mais pourtant faire des dégâts à cause de la latence en multijoueur. Cependant, si la position de la balle était synchronisée en ligne, cela consommerait une telle bande-passante que la latence serait encore plus élevée.

4.6.15 Débogage

Une très grande partie de mon temps de travail a été dédiée à la correction de bogues notamment avec l'arrivé de l'IA et du système de combat et d'autant plus lorsqu'il s'agit du multijoueur.

4.6.16 Améliorations diverses

Beaucoup de détails ont été améliorés ou ajoutés. Ils ne sont ni indispensable ni très compliqués, il s'agit en général de messages d'aide dans les menus au cas où le joueur n'aurait par exemple pas mis de nom pour sa partie ou bien s'il ouvre le panneau des alliés, ou encore un petit message « *waitting for players...* » dans la salle d'attente pour rappeler au joueur qu'il est bien dans la salle d'attente ainsi que son but.

4.6.17 Travail en groupe

Nos tâches ayant été bien réparties, nous pouvons travailler un maximum chacun de notre côté sans avoir besoin des uns ou des autres. Nous organisons régulièrement des réunions pour faire des points sur ce qui a été fait et reste à faire ainsi que pour discuter de la direction que va prendre le jeu, on débat de la manière d'implémenter certaines fonctionnalités, en somme, on définit ce que l'on veut que le jeu soit.

Concrètement cela fonctionne plutôt bien il n'y a jamais eu de différents majeurs et nos différents domaines d'expertise nous permettent de nous faire confiance les uns les autres.
D'un point de vue technique nous échangeons sur le logiciel Discord et nous avons initialisé un dépôt sur GitHub.

4.6.18 Optimisation

Pour la dernière soutenance mon travail a notamment été d'optimiser le jeu dans l'optique qu'il puisse être lancer par un maximum d'ordinateur différent avec des spécifications techniques variable et notamment les moins puissants.
Le jeu était bien optimisé jusqu'alors et pouvait tourner sur beaucoup de machines convenablement. Cela s'explique par le fait que le jeu n'utilise pas d'effets graphiques particulièrement gourmands et ceux qui le sont le plus comme les ombres, l'occlusion ambiante et l'anticrénelage se désactive si on baisse les options graphiques du jeu.
De manière générale les jeux de type RTS demandent une configuration relativement légère par défaut. Cependant c'est un type de jeu qui ne se prête pas très bien à des configurations fixes. En effet le nombre d'unités qui peuvent exister à la fin d'une partie peut devenir très élevés, et d'autant plus s'il y a beaucoup de joueurs dans la partie. Ainsi ce genre de jeu passe de très peu exigeants en termes de ressources à très exigeant lorsque le nombre d'unité est conséquent.
Nôtre jeu n'échappe pas à ce problème et ce à cause d'un seul élément du jeu qui est très exigeant et qui est la détection d'autres unités. Chaque unité d'un joueur possède un rayon de détection qui sert à afficher ou non les unités ennemies (voir brouillard de guerre) ou bien les attaquer.
A partir d'environ 80 unités à l'écran, le nombres d'image par seconde sur un ordinateur équipé d'un processeur moyenne-gamme descendra en dessous de 60. Cependant, si ces mêmes unités sont plus éparpillées et non les unes sur les autres alors les performances remontent très vite. Il est malgré tout très rare de dépasser ne serait-ce que 50 unités sur un même écran, ainsi ce cas de figure est très rare et dans tous les autres cas le jeu tourne très bien sur des configurations très légères.
J'ai tout de même cherché à optimiser cela, et c'est ainsi que je me suis rendu compte grâce au « profiler » d'Unity (outils permettant de voir ce qui consomme de le plus de ressource dans notre jeu) que les composant unité pour détecter la collision est bien plus performant en utilisant des sphères que des boites (rectangle en 3 dimensions). L'amélioration est légère mais présente. De plus la meilleure optimisation que j'ai pu faire a été de remplacer mon système de col-

lision par un autre. Jusqu'alors je me servais des composant de Unity nommés « colliders » qui permettent très simplement de détecter lorsqu'un autre collider entre en contact avec lui, lorsqu'il en sort et lorsqu'il y reste. Cependant je n'ai pas besoin de détecter lorsqu'un élément entre dans la zone mais seulement lorsqu'il y reste ou bien en sort. Des calculs inutiles alourdissaient alors le jeu. De plus pour que les colliders puissent se détecter les uns les autres, l'objet qui les contenait se devait d'avoir aussi un « rigidbody » qui sert à calculer la physique d'un objet. Or dans un jeu de stratégie dans l'espace il est inutile de calculer la physique (toutes les unités restant à la même hauteur dans le jeu et ne rentrant pas en collision directement). J'ai ainsi trouvé une autre méthode qui s'appelle « overlapsphere » qui permet de se passer de rigidbody et qui ne détecte que les unités qui sont dans le rayon de collision et non lorsqu'elle entre ou en sorte. Avec un algorithme simple je peux savoir lorsqu'elle en sorte. Mais ce nouveau système à bien amélioré les performances du jeu.

Après optimisation nous avons pu tester qu'avec un processeur moyenne gamme (Inter Core i5-7300HQ) le jeu pouvait supporter 150 unités avant de voir son nombre d'image par seconde baisser en dessous de 60 et 220 unités pour un processeur haut de gamme (i7-7700).
Etant donné que la population maximum par base est de 100 et que les unités de combat coutent entre 2 et 10 points de population, on peut considérer qu'un armé complet de fin de jeu possède environ 30 à 40 unités maximum et donc se retrouver avec plus de 150 unités dans une zone de jeu très petites demanderait au minimum 4 joueurs, ce qui est le maximum de joueur possible durant une partie.
De plus ces résultats sont issus de test effectué avec les paramètres graphiques au maximum, au minimum les performances sont environ 15% plus élevées.

4.6.19 Configuration Minimale et Recommandée

Configuration Minimale :
OS : Windows 7,8 ou 10
Processeur : Intel ou AMD Dual-Core à 1.70Ghz
Mémoire : 2Go
Stockage : 2Go
Carte Graphique : carte NVIDIA ou AMD compatible DirectX 11
avec 1.5Go de mémoire

Configuration Recommandée :

OS : Windows 7,8 ou 10
Processeur : Intel ou AMD Dual-Core à 1.70Ghz
Mémoire : 2Go
Stockage : 2Go
Carte Graphique : NVIDIA GT1040 ou plus | AMD RX 540 ou plus

Tests effectué grâce au logiciel de Benchmark MSI Afterburner.

4.6.20 Bilan

Pour la première soutenance j'ai développé un maximum de fonctionnalités pour le jeu tout d'abord parce qu'il me fallait faire une Intelligence artificielle pour la prochaine et que je n'ai aucune expérience dans ce domaine mais aussi pour qu'entre la deuxième soutenance et la dernière je n'ai pour tâche que peaufiner le jeu, l'optimiser, le déboguer.Notre but a été de fournir un jeu atteignant un certain standard de qualité.Nous voulions que n'importe qui puisse voir notre jeu et penser qu'il s'agit d'un véritable jeu destiné à la commercialisation et non un jeu à but éducatif développé en 6 mois par des 4 élèves. Nous pensons que ce standard a été atteint.

5 Conclusion

Après de nombreuses heures de travail, notre jeu est enfin finalisé et toutes les fonctionnalités ont été implémentées. Nous avons su travailler dans le détail du jeu et le peaufiner, et c'est donc avec fierté que nous vous présentons, pour cette troisième soutenance, Year Zero.