Project : HGF

Rapport de projet

soutenance finale, le 17 juin 2008

Félix *Flx* **Abecassis** (*abecas_e*)
Christopher *Vjeux* **Chedeau** (*chedea_c*)
Vladimir *Vizigrou* **Nachbaur** (*nachba_v*)
Alban *Banban* **Perillat-Merceroz** (*perill_a*)

Table des matières

1 *Introduction* 5

2 *But et intérêts du projet* 6
2.1 Qu'est ce qu'un RTS ? . . . 6
2.2 *Environnement du jeu* . . . 7
2.3 Intérêts . . . 8
2.3.1 Recherches personnelles . . . 8
2.3.2 Délais . . . 8
2.3.3 Travail en équipe . . . 8

3 *Historique de la Fooo Team* 9
3.1 Formation du groupe . . . 9
3.2 Première Soutenance . . . 10
3.3 Deuxième Soutenance . . . 11
3.4 Troisième Soutenance . . . 12
3.5 Soutenance Finale . . . 13

4 *Alban : Communication, Réseau, Son* 14
4.1 Mon rôle dans le groupe . . . 14
4.1.1 Réseau . . . 14
4.1.2 Son . . . 15
4.1.3 Communication . . . 15
4.2 Récit de réalisation - Communication . . . 16
4.2.1 Le site web . . . 16
4.3 Epipub . . . 17
4.3.1 Marketing . . . 17
4.4 Récit de réalisation - Réseau . . . 18
4.4.1 Premiers pas . . . 19
4.4.2 Abandon d'Indy . . . 19
4.4.3 Intégration du réseau dans le jeu . . . 20
4.4.4 Simplification de la procédure de connection . . . 20

4.4.5 Intégration au menu 21
4.4.6 Synchronisation des données 23
4.5 Récit de réalisation - Son 24

5 *Christopher : Moteur Graphique et Interface Utilisateur* 26
5.1 Mon rôle dans le groupe 26
5.2 Enjeux et difficultés 26
5.2.1 Enjeux 26
5.2.2 Difficultés 27
5.3 Récit de réalisation - Moteur Graphique 28
5.3.1 Affichage du terrain 28
5.3.2 Déplacement de la Caméra 28
5.3.3 Affichage du modèle 30
5.3.4 Optimisation 30
5.3.5 Picking 32
5.3.6 Animations 32
5.3.7 Améliorations 34
5.4 Récit de réalisation - Interface Utilisateur 35
5.4.1 Fonctionnement 35
5.4.2 Texte Affiché 38
5.4.3 Cadres 39
5.4.4 Contrôles 39
5.4.5 Minicarte 41
5.5 Mes impressions 42
5.5.1 Moteur Graphique 42
5.5.2 Interface Utilisateur 42
5.5.3 Travaux Annexes 42
5.5.4 Conclusion 43

6 *Vladimir : Déplacement des Unités* 44
6.1 Mon rôle dans l'équipe 44
6.2 Déplacer des unités 45
6.3 Enjeux et difficultés 45
6.4 Première soutenance 46
6.5 Deuxième soutenance 46
6.5.1 Recherche de chemin 47
6.5.2 Système d'ordres 48
6.6 Troisième soutenance 48
6.6.1 Recherche de chemin 48
6.6.2 Ordres et mouvements 49
6.7 Soutenance Finale 50

6.8 Mes impressions 50

7 *Félix : Gestion des données et interface* 52
7.1 Mon rôle dans le groupe 52
7.2 Présentation des mécanismes du jeu 53
7.3 Enjeux et difficultés 54
7.3.1 Enjeux 54
7.3.2 Difficultés 55
7.4 Récit de réalisation 56
7.4.1 Première Soutenance 56
7.4.2 Deuxième Soutenance 59
7.4.3 Troisième Soutenance 62
7.4.4 Soutenance Finale 65
7.4.5 Mes impressions 67

8 *Conclusion* 68

Annexes 69

Chapitre 1

Introduction

Bonjour et bienvenue à la découverte du Project : HGF réalisé par la Fooo Team. C'est avec plaisir que nous allons vous montrer comment quatre jeunes étudiants de première année d'EPITA, après 8 mois de travail acharné, ont réussi à créer un jeu vidéo.

Vous decouvrirez toutes les étapes nécessaires pour qu'à partir d'une petite idée émerge un jeu pleinement fonctionnel, intuitif et visuellement beau. Les amateurs de détails croustillants sur l'aventure seront ravis quant aux personnes qualifiées sur le sujet, elles vont pouvoir profiter de tous les détails techniques.

A vous maintenant de vous plonger dans l'univers Fooo.

Chapitre 2

But et intérêts du projet

2.1 Qu'est ce qu'un RTS ?

Les RTS sont régis par certaines règles basiques qui permettent d'inscrire un jeu dans ce type. Il s'agit tout d'abord d'un jeu de stratégie multijoueur, dans lequel chacun doit gérer une communauté, voire une civilisation, en lui donnant des ordres. Le joueur doit peu à peu accumuler des ressources en développant des techniques de récolte et autres afin de créer des bâtiments et des unités de combat. Le but final est d'être meilleur que les autres joueurs, en les faisant capituler ou en les éradiquant jusqu'au dernier par la force.
Ce qui caractérise un tel jeu est bien sur la liberté de se développer comme on le souhaite, en construisant certains bâtiments, en recherchant des améliorations spécifiques et en décidant du moment propice pour attaquer ses ennemis.

Pour pouvoir le qualifier de "stratégique", il est donc nécessaire que le joueur se retrouve confronté à des choix au cours de la partie concernant son orientation économique et militaire, tout en faisant en sorte qu'il n'y ai jamais une seule méthode qui marche à chaque fois. C'est l'un des points forts de ce type de jeu, mais aussi une difficulté supplémentaire pour le projet concernant l'équilibrage des unités.

FIG. 2.1 – Command & Conquer : Conflit du Tibérium, sorti en 1996

2.2 *Environnement du jeu*

Notre jeu se déroule dans un univers médiéval-fantastique et met en scène deux races différentes : les rats et le tréants, en guerre depuis des temps immémoriaux, comme décrit dans la partie Histoire du manuel de jeu. Pour se développer, le joueur doit amasser deux types de ressources, la première est l'or qui s'acquiert en envoyant des unités en mode extraction dans les bâtiments dédiés et la deuxième est le nombre d'unités ennemies tuées. Ces ressources sont nécessaires pour la construction de bâtiments et pour la formation d'unités plus évoluées. Le but est toujours d'éliminer l'autre, sans négliger sa propre base, car la perte de tous ses bâtiments engendre la défaite immédiate.

2.3 Intérêts

Nous étions conscients de la difficulté et nous possédions une idée assez précise du travail que pouvait représenter projet de cette envergure. Cependant cet aspect de défi a rendu le projet plus stimulant et enrichissant pour nous tous. Il nous a été en effet nécessaire d'aller au-delà des bases que les cours nous donnent, nous avons appris à chercher par nous même comment contourner certains problèmes, et à trouver des solutions simples et efficaces.

2.3.1 Recherches personnelles

Un ingénieur informaticien se doit d'être autodidacte pour s'adapter aux nouvelles technologies en évolution perpétuelle. Le but recherché était donc d'acquérir le réflexe de rechercher par nous-même toutes les informations, qualité indispensable pour mener à bien des projets comme celui que nous avons choisi.

2.3.2 Délais

Nous avons appris à tenir des délais, à nous conformer au cahier des charges, en effet, il était difficile de prévoir si les délais seraient respectés lors de la phase de réflexion sur les mécanismes et aspects du jeu. Lors de cette dernière soutenance, nous estimons avoir accompli ce que nous espérions : obtenir un produit fini et jouable.

2.3.3 Travail en équipe

Nous avons également appris à travailler en équipe, à prendre des décisions communes, ou encore à nous motiver mutuellement en cas d'obstacle majeur rencontré par l'un des membres du groupe. Les différences de niveau et d'expérience initiales entre nous ont rajoutées des points de réflexion, nous avons dû réfléchir longuement à la répartition de certaines tâches dans un souci d'équité, mais aussi dans le respect des compétences et préférences de chacun.

Chapitre 3

Historique de la Fooo Team

3.1 Formation du groupe

Notre groupe s'est rapidement formé, avant même le début des inscriptions officielles sur le site de Nathalie Bouquet, en effet nous étions pleinement conscients qu'un projet de cette envergure et de cette ambition représente un travail gigantesque. Cette convergence d'esprits s'est réalisée sous l'impulsion de nos affinités communes et de notre motivation immuable de créer un jeu de qualité.

Une fois le groupe constitué, nous avons réfléchis ensembles pour décider de la nature de notre jeu, sur les 4 membres constituant l'équipe, trois d'entre nous sont des amateurs de Warcraft 3, nous avons arrêté notre choix sur la création d'un jeu de stratégie en temps réel en suivant ce modèle. Une fois ce choix effectué, nous nous sommes rassemblés dans les locaux de l'EPITA pour distribuer les taches et réfléchir à une ébauche de cahier des charges.

Ensuite, nous nous sommes à nouveau retrouvés afin de réfléchir de manière théorique sur les différents aspects du jeu tel que le réseau, la gestion et la synchronisation des données, la recherche de plus court chemin, etc.

Contrairement à Félix et Alban, Christopher et Vladimir possédaient déjà une solide expérience de la programmation, ces derniers ont donc eu en partie au début la tâche de nous former et de nous conseiller sur certains aspects.

3.2 Première Soutenance

La première étape du développement du projet a été le choix du support de développement, concernant la possibilité entre Delphi et Caml, nous avons rapidement décidé d'utiliser le Delphi pour sa facilité de prise en main, la puissance de son interface de développement et la documentation assez abondante sur Internet. Le choix plus difficile a été celui de la bibliothèque graphique entre DirectX et OpenGL, nous avons finalement choisi ce dernier car il fonctionne généralement sur toutes les machines ce qui a rendu possible le travail sur le projet aussi bien chez nous que en salle machine à l'EPITA.

Nous avons commencé le travail sur le projet très tôt mais la finalité recherchée dépendait du niveau de chacun et des tâches attribuées. Christopher a immédiatement commencé l'élaboration d'un loader de modèles. Félix s'est concentré sur l'apprentissage du Delphi et sur une réflexion de choix de modèle de gestion de données. Alban, en plus d'une ébauche de réseau, a effectué d'entrée un important travail de marketing avec la réalisation du site web dans sa quasi totalité mais aussi avec l'impression de logos Fooo sur des chemises blanches que nous avons dès lors portées à chaque soutenance pour mettre en valeur notre appartenance et notre attachement à notre équipe de projet. Vladimir quant à lui a effectué des recherches théoriques sur le fonctionnement de l'algorithme A* et a entamé une simulation de résolution de recherche de plus court chemin.

Lors de cette première soutenance, les réflexions théoriques étaient communes mais le développement était individuel, il était en effet trop tôt pour rassembler notre travail, le temps était plutôt à l'expérimentation chacun de notre côté avant de penser à une quelconque convergence. Afin de travailler efficacement et dans le but de préparer tous les aspects du déroulement de la soutenance, nous nous sommes réunis durant trois jours chez Alban au Kremlin-Bicêtre.

3.3 Deuxième Soutenance

Deux mois après la première soutenance, la Fooo Team est passée à nouveau devant le jury pour présenter l'avancée de son projet de jeu de stratégie. Durant ce court de laps de temps occupé par de nombreux cours et contrôles d'autres matières, le travail sur le projet s'est axé majoritairement sur la convergence de tous nos travaux de la première soutenance, la mise en place d'un système de gestion de versions (SVN) a été l'élément déclencheur de ce regroupement.

Mis en place sur le serveur dédié qui nous servait déjà à héberger le site web, ce SVN nous a permit de travailler sur l'ensemble du projet, aussi bien à distance que tous regroupés dans une même pièce de 10m2 comme cela a été le cas durant la semaine précédent la soutenance, nous nous sommes réunis dans l'appartement de Christopher dans Paris pour coder efficacement et rapidement en coopération dans une bonne atmosphère générale. Nous étions bien installés, en séparant zone de travail et zone de repos, et ce rapprochement a permis d'avancer efficacement dans le développement en gardant toujours le même objectif d'obtenir un jeu fini, jouable et intuitif.

C'est un véritable travail de groupe qui a donc pris le dessus sur les expérimentations individuelles de la soutenance précédente, notamment grâce à une ambiance et une entente parfaite, conditions indispensables pour surmonter les diverses difficultés qui nous ont barré la route.

3.4 Troisième Soutenance

Les deux mois qui suivirent la deuxième soutenance furent très prolifiques pour l'avancement du projet HGF. Nous avions réuni lors de celle ci toutes nos parties au sein d'un même programme grâce à un important travail d'équipe. Nous avons continué dans ce sens dans un premier temps afin de poser définitivement les bases du jeu. C'est ensuite la jouabilité qui a guidé notre développement. Qui dit jouabilité dit interaction, le développement de l'interface à donc joué un rôle central durant cette préparation de soutenance.

Notre réunion s'est cette fois déroulée dans l'appartement de Vladimir à Antony, nous étions encore une fois dans de bonnes conditions car nous avions à notre disposition beaucoup de place et que grâce à la séparation entre lieu de travail et de repos, il était possible pour n'importe qui d'aller dormir sans risquer d'être dérangé par les autres membres. Des groupes de travail ont naturellement émergé afin de se mettre d'accord sur certaines parties communes. On peut citer Alban, Christopher et Félix qui ont su communiquer afin d'apprendre une structure générique d'interface nouvelle pour tous, mais aussi Alban, Félix et Vladmir qui ont coordonner leurs efforts pour tout ce qui concerne les ordres des unités des unités et la synchronisation des données par le réseau. De nombreuses améliorations et fonctionnalités ont également étés intégrées afin d'obtenir un jeu optimisé, performant et intuitif. Nous étions à cette période toujours motivés et solidaire dans ce projet où sont survenus de nombreux obstacles et difficultées.

3.5 Soutenance Finale

Pour la soutenance finale l'objectif était bien sur que tout soit terminé et fonctionnel. Durant la semaine allouée Alban et Christopher se sont réunis dans un premier temps, puis nous avons passé les trois derniers jours chez Vladimir afin de tout finaliser.

Nous nous sommes réuni au sous sol des associations au Kremlin Bicêtre, dans les locaux généreusement prêtés par EpiRadio, afin d'enregistrer les voix des unités, il a fallu ensuite chacun de notre coté réaliser le traitement et le découpage des fichiers sons ainsi obtenus afin qu'ils soient utilisables.

Nous avons aussi réfléchis via le wiki que nous avons mis en place sur les différents éléments à rendre pour la soutenance finale. Nous avons également défini les unités, bâtiments et leurs caractéristiques propres.

Chapitre 4

Alban : Communication, Réseau, Son

4.1 Mon rôle dans le groupe

Je suis chargé dans le projet HGF du réseau, du son et de tout l'aspect communication lié au projet.

4.1.1 Réseau

Le réseau a été la majeure partie, l'objectif étant de pouvoir s'affronter à plusieurs dans le jeu.

Enjeux

Le réseau est un élément clé du jeu puisque celui-ci est basé uniquement sur le mode multijoueur. Il fallait donc impérativement que celui-ci soit fonctionnel et efficace.

Difficultés

Les principales difficultés que j'ai rencontrées étaient liées au réseau : j'ai beaucoup tâtonné pour trouver le bon outil, et une fois celui-ci fixé définitivement, il a fallu trouver les bonnes méthodes pour l'utilisation spécifique que j'en fais. Le jeu de stratégie en temps réel est en effet un des systèmes les plus complexes à synchroniser, justement à cause de son aspect "temps réel". En plus de la synchronisation, la difficulté a aussi résidé dans la quantité importante de données à faire transiter.

4.1.2 Son

Le moteur de son devait pouvoir jouer aussi bien des musiques que des échantillons de son plus courts.

Enjeux

Les sons sont importants pour l'immersion dans le jeu. Il fallait donc un moteur capable de les lire, mais aussi sélectionner et enregistrer des sons de qualité.

Difficultés

Le moteur de son en lui-même n'a pas posé de difficultés majeures. Le plus difficile a été au final de trouver des musiques à la fois cohérentes avec l'ambiance que nous visions et libres de droits.

4.1.3 Communication

Enfin l'aspect communication consiste en la mise en place du site web, ainsi que tous les produits dérivés qui permettent d'assurer une bonne image à notre projet.

Enjeux

La communication ne parait pas si importante au premier abord. Cependant le site web est la seule interface entre le projet et la communauté de joueurs potentiels. Il est donc primordial de le soigner afin d'intéresser le plus de monde possible. Celui-ci présente le projet, l'équipe de production, permet de télécharger le jeu et les différentes ressources liées au projet, mais permet aussi au lecteur de s'informer des dernières nouvelles sur l'état d'avancement du projet, la sortie de patches correctifs, etc.

Le deuxième aspect de la communication a été de soigner l'image de la Fooo Team à la manière d'une marque : il fallait donc trouver un visuel accrochant et simple à la fois.

Difficultés

La réalisation du site web n'a pas vraiment été difficile, mais mes faibles connaissances en matière de programmation web m'ont obligé à tout apprendre sur le coup, ce qui a rendu la tâche plus longue que prévu.

4.2 Récit de réalisation - Communication

4.2.1 Le site web

Conception

La réalisation du site web fut pour moi une expérience très enrichissante. Je n'en étais pas à mes premiers pas en matière de réalisation de sites web, mais je n'avais jamais vraiment réalisé le mien de A à Z. Mon but était de réaliser un site à la fois simple et « propre » au niveau de sa conception. Le respect d'une sémantique et des standards du web sont restés pour moi une priorité.

Le développement de ce site m'a fait découvrir différents langages : le xHTML pour la sémantique, le CSS pour la mise en forme, le PHP pour la structure dynamique des différentes pages et localisations, et enfin le XML pour stocker le texte brut, selon la langue désirée.

Ce site est en effet localisé selon la langue de l'utilisateur : un fichier XML différent est appelé en fonction de la langue choisie. Il est en effet essentiel de disposer d'une version anglaise du site web afin d'élargir sensiblement le public visé.

Le site web a été mis en ligne officiellement au début de l'année 2008 à l'adresse http ://www.fooo-team.com.

Le site a été mis à jour continuellement au fur et à mesure de l'avancement du projet, et a de plus subit deux modifications majeures : la mise en ligne de la version anglaise, puis des versions espagnoles et allemandes, respectivement lors de la deuxième et la troisième soutenance. Ces modifications ont seulement nécessité un effort de traduction, puisque le système de stockage des informations au format XML ne nécessite pas de modification du code source du site pour ajouter une nouvelle version.

Sans véritablement approfondir, la réalisation de l'interface de changement de langue m'a permis de découvrir le JavaScript, en plus de toutes les autres technologies précédemment citées.

Référencement

Afin de simplifier l'accès au site du projet, nous avons saisi l'opportunité de la libération du nom de domaine fooo.fr pour le racheter. Il est maintenant possible de se connecter sur le site grâce à l'adresse http ://www.fooo.fr, qui est beaucoup plus simple à retenir.

Afin de s'assurer une meilleure visibilité auprès du grand public il était nécessaire de soigner le référencement du site web. Il a fallu pour cela réaliser le site en respectant une certaine sémantique, reconnue plus facilement pas

les robots des moteurs de recherche, et déposer des liens vers notre site sur tous les sites que nous pouvions. C'est une des raisons de la création du projet EpiPub.

4.3 Epipub

EpiPub est un projet que j'ai réalisé en partenariat avec Jean-Romain Prevost, du groupe « Freedom Fries ». Il s'agit d'une petite régie publicitaire destinée à créer un lien entre les différents sites de projets d'Epita, et plus largement tous les sites développés par les étudiants d'Epita, ayant plus ou moins de rapport avec l'école.

Il consiste en un échange de bannières entre les différents sites, affichées aléatoirement. Ainsi pour s'inscrire, il suffit de fournir trois bannières de tailles standards : 468x60, 120x300 et 180x150, au format jpg, png, gif ou même flash grâce au formulaire disponible sur le site créé pour l'occasion : http ://www.epipub.info. Le dossier est ensuite validé manuellement pour éviter les éventuels abus. Une fois le dossier validé, le site en question n'a plus qu'à afficher au moins une des bannières sur son site grâce aux trois liens que nous lui fournissons, correspondant chacun à une des trois bannières.

Le projet a connu un bon succès. Quinze projets ont en effet adhéré à la régie.

FIG. 4.1 – La bannière Fooo sur EpiPub

4.3.1 Marketing

La plus grande réussite marketing de ce projet, outre le fait d'avoir été en haut du tableau des notes, a été le choix du nom du groupe et la réalisation d'un logo à la fois simple, original et accrocheur.

J'ai donc réalisé des fonds d'écran qui affichent fièrement le logo. Le fait de les utiliser sur les machines de l'EPITA par exemple a permis de faire connaître l'équipe à un grand nombre d'élèves.

De plus, une vidéo d'introduction nous a permis d'imposer notre image d'une façon encore plus frappante.

Afin de donner une image professionnelle au jury du projet, nous avons fait faire des chemises à l'effigie de la Fooo Team. Elles nous permettent de nous présenter de façon uniforme lors des soutenances, tout en affichant encore une fois le logo Fooo. C'est le Studio Chaillot[1] que j'ai retenu pour réaliser cette tâche.

Afin de permettre à notre jury de soutenance de profiter au maximum de la deuxième présentation, nous lui avons permis de déguster un café dans les meilleurs conditions possibles : une tasse à l'effigie de la Fooo Team. Réalisée spécialement pour l'occasion par notre imprimeur habituel.

Après la sortie du jeu

C'est cependant pendant et après la sortie du jeu qu'il va falloir attirer le plus de monde possible pour télécharger notre jeu. Pour cela il va falloir faire connaître notre jeu : nous pourrons le présenter dans des forums spécialisés, tenter de faire parler le plus possible de nous sur d'autres sites.

Il sera nécessaire d'apporter un support technique vis-à-vis du jeu et des retours que nous aurons : nous n'avons pas la prétention de sortir un jeu exempt de tout bug, et nous sortirons de nouvelles version au fur et à mesure de leur correction.

Nous avons également le plaisir d'annoncer que le projet sera présenté au Festival du Jeu Vidéo 2008, se déroulant du 26 au 28 septembre 2008 à Paris Expo, Porte de Versailles. Christopher et moi-même avons proposé à l'école d'organiser la mise en place d'un stand EPITA dans ce salon, afin de présenter l'école via ses jeux-vidéos réalisés en première année. Toute la Fooo Team sera donc présente pour animer le stand, présenter trois projets sélectionnés parmi les meilleurs, dans lesquels sera présent Project : HGF. La participation à ce salon assurera la promotion de l'EPITA mais aussi de notre projet auprès des visiteurs passionnés de jeux vidéo.

4.4 Récit de réalisation - Réseau

Après avoir réalisé la majeure partie du site web, je me suis consacré à ma tâche principale dans ce projet : la réalisation d'un système de réseau permettant à plusieurs joueurs de s'affronter dans des parties homériques.

L'idée est tout d'abord de créer un système de serveur, auquel pourrait se connecter un nombre indéterminé de clients, échangeant de simples chaînes de caractères. Ce qui pourrait s'apparenter à un simple « chat » se transformera,

[1] Studio Chaillot, Paris 16^{e}, http ://www.shirt-photo.com

une fois ce texte interprété comme des commandes de jeu, en un centre de synchronisation des différents joueurs, en clair, un moteur réseau.

4.4.1 Premiers pas

Le développement du réseau a de nombreuses fois été tourmenté. J'ai en effet beaucoup tâtonné pour trouver les bons outils, et pour ensuite trouver une manière optimale de les utiliser.

J'ai commencé à utiliser l'outil « TForm » de Delphi pour faire deux fenêtres simples, un serveur et un client, avec chacune une boîte de texte qui affiche les dernières actions effectuées. J'ai tout d'abord utilisé les composants internes de delphi : « Ttcpserver » et « Ttcpclient ». Ceux-ci ont rapidement montré leurs limites, je me suis donc rabattu sur une librairie. J'avais connaissance de deux librairies réseau en delphi : ICS et Indy. J'ai choisit Indy pour son intégration à Delphi 2007.

4.4.2 Abandon d'Indy

L'utilisation d'Indy s'est rapidement averée compliquée pour moi : la documentation s'est avérée inefficace et la multitude de composants disponibles ne faisait que de noyer les outils que j'utilisais vraiment.

J'ai donc choisis d'utiliser directement le composant Winsock, intégré à Windows. Celui-ci comporte plusieurs avantages à mes yeux : sa légèreté le rend simple d'utilisation et me permet de comprendre le fonctionnement de chaque fonction que j'utilise. Sa portabilité me permettra par la suite de

réutiliser mes connaissances acquises dans différents langages et sur différentes plateformes. Enfin ce composant est très bien documenté, autant dans sa version Windows que Unix.

4.4.3 Intégration du réseau dans le jeu

Après avoir rapidement posé les bases du réseau dans une application console, je me suis attaqué notamment grâce au SVN à l'intégration du réseau dans le jeu. J'ai pu dans un premier temps tester l'envoi de messages grâce à l'affichage de texte sur la gauche de la fenêtre de jeu. Par la suite, l'arrivée de la saisie de texte m'a permis de mettre en place un véritable chat permettant d'envoyer du texte personnalisé.

Le chat étant opérationnel, j'ai du m'intéresser à la structure du réseau : les données étaient pour l'instant directement envoyées selon une adresse IP, et il fallait mettre en place un système de serveur auquel les clients se connectent, pour centraliser les informations. Contrairement à ce que je pensais, il fut plus simple de faire un serveur intégré au client et non dédié. Les joueurs choisissent un serveur auquel se connecter, et celui-ci est chargé de recevoir toutes les informations de chaque joueur et de les diffuser aux clients intéressés.

Pour envoyer des données autres que du texte, il m'était intéressant de pouvoir envoyer des types structurés par le réseau. Le composant Winsock le permet très facilement : je peux ainsi identifier le type de données reçues par le serveur à l'aide du premier champ du type structuré et donc le traiter en conséquence.

Il m'a donc été possible, en plus du chat, d'envoyer l'ordre de création d'une unité par le réseau. Les unités affichées à l'écran d'un joueur sont distinguées par un code de couleur : les unités bleues sont celles du joueur, les rouges sont celles des joueurs adverses. Il est aussi possible d'infliger des dégâts à une unité via le réseau, et ainsi la faire disparaitre si celle-ci venait à mourir.

4.4.4 Simplification de la procédure de connection

Je me suis ensuite penché sur la procédure de connexion à un serveur. Il était auparavant nécessaire de connaitre l'adresse IP de l'hôte ainsi que la sienne pour se connecter à l'aide d'une ligne de commande intégrée au jeu. Le but étant à terme de pouvoir se connecter à un serveur sans connaitre ces deux informations, je me suis donc intéressé à la technique de broadcasting. Ceci consiste à envoyer un même paquet à une plage d'IP donnée. Dans notre cas, ce paquet sera envoyé à toutes les adresses IP locales. Si un serveur est

lancé, il recevra ce paquet et pourra indiquer sa présence au client cherchant un serveur. Il n'est donc plus nécessaire de connaitre aucune adresse IP lors de la procédure de connexion.

4.4.5 Intégration au menu

Il fallait maintenant pouvoir se connecter à un serveur existant, ou en créer un de façon claire et simple. Grâce au travail de Christopher sur le XML et le Lua, il m'a été possible de scripter les pages suivantes :

Rejoindre une partie

La page permettant de rejoindre une partie se décompose en deux parties : une première pour se connecter directement en connaissant l'adresse IP du serveur (nécessaire à une connexion via internet), et une deuxième partie pour rechercher un serveur sur un réseau local : les parties sont automatiquement trouvées grâce à la technique de broadcasting, précédemment expliquée. Les informations du serveur sont affichées sous forme de liste, avec notamment le nom du serveur, son adresse IP, le nombre de joueurs actuellement connectés sur ce serveur et la carte de jeu. Un bouton est disponible pour rafraichir la liste des serveurs, et un autre pour se connecter au serveur sélectionné.

FIG. 4.2 – Rejoindre une partie en réseau

Créer une partie

L'écran de création d'une partie permet de choisir le nom du serveur, son propre nom, et de sélectionner une carte. Les informations sur la carte sélectionnée sont affichées sur la droite.

Attendre les autres joueurs

Enfin, une page qui permet aux joueurs connectés sur un serveur de patienter en attendant le début de la partie : elle affiche les joueurs présents sur le serveur et se rafraichit automatiquement à chaque nouveau joueur entrant sur le serveur pour mettre à jour la liste. Le joueur qui héberge la partie peut à tout moment lancer la partie grâce à un bouton dont lui seul a l'unique accès. Une fois la partie lancée, le serveur n'apparaitra plus dans la liste des serveurs disponible : en effet dans un jeu de stratégie, aucun joueur ne peut rejoindre une partie déjà lancée.

FIG. 4.3 – Joueurs en attente dans un serveur

4.4.6 Synchronisation des données

La synchronisation des différents joueurs est la dernière chose sur laquelle je me suis penché, mais certainement aussi la plus difficile. Il a fallu dans un premier temps régler les horloges de tous les joueurs à la même heure, pour que chaque action se déroule en même temps. Pour cela j'ai mis en place un système qui mesure le temps de latence de chaque joueur, et je peux donc sur plusieurs mesures trouver le temps de latence moyen d'un client par rapport au serveur. C'est grâce à ces calculs qu'il est possible de synchroniser tous les clients entre eux.

Afin d'obtenir une synchronisation acceptable, j'ai procédé de différentes manières selon le type de donnés à partager :

Synchronisation par événement

Pour les données de type ressources (l'or, le bois, mais aussi les points de vie de chaque unités par exemple), les données du serveur sont envoyées à chaque modification. Si les points de vie d'une unité sont modifiés, le serveur en informe les différents clients. L'ancienne valeur que possédait le client est écrasée : cette redondance permet de prévenir d'éventuels problèmes réseaux.

Ordres

Les clients envoient au serveur les informations sous formes d'ordres, ou plutôt de requêtes. Pour se déplacer, un client indique au serveur l'unité concernée et la position d'arrivée. Le serveur se charge ensuite de lui transmettre les nouvelles positions des joueurs.

Déplacements

Le déplacement a posé quelques problèmes au niveau de la synchronisation. Un premier système mis en place a été la gestion des déplacements

par un système d'ordres du serveur vers le client. Ce mode de gestion n'a pas du tout fonctionné puisque deux ordinateurs différents ne prennent pas autant de temps pour calculer un même chemin : les joueurs n'étaient pas synchronisés et les positions finales n'étaient parfois pas les mêmes.

Il a donc fallu choisir autre moyen de gérer le déplacement tout en s'assurant que les données sont les mêmes chez les différents clients.

La méthode retenue est la suivante : le serveur calcule le chemin et déplace l'unité. Il envoie périodiquement la position de l'unité et la direction dans laquelle elle se dirige. Ainsi chaque client positionne l'unitée et la fait se déplacer en attendant une nouvelle position donnée par le serveur. On obtient un déplacement fluide, et surtout le même résultat chez tous les joueurs.

4.5 Récit de réalisation - Son

Je me suis occupé de l'implémentation technique du son dans le jeu.

Après quelques recherches, il a semblé judicieux de retenir la bibliothèque FMod[2] : celle-ci est en effet gratuite pour des utilisations non commerciales, et s'avère très bien adaptée à des jeux, et de qualité professionnelle.

Techniquement, le son peut être géré globalement de deux manières différentes :

- **Le sample** : adapté aux sons de petite taille et fréquemment utilisés. Le son est préchargé dans la mémoire vive. Le seul désavantage est la quantité de mémoire utilisée : ce mode est donc uniquement adapté aux sons courts.

- **Le streaming** : cette fois destiné aux sons de plus grande taille, tels que la musique : il n'est ici pas question de préchargement dans la mémoire vive : le son est directement lu depuis le disque dur, ce qui nécessite un accès au disque et donc demande plus de ressources au système. Ce mode est utilisé pour les musiques, puisqu'il n'est pas question de précharger plusieurs morceaux d'un ordre de grandeur de cinq mégaoctets chacun directement dans la mémoire vive.

[2] http ://www.fmod.org

Mis à part les musiques d'ambiance, le mode le plus utilisé sera le mode sample : il est en effet nécessaire dans un jeu de stratégie de jouer fréquemment de courts sons : les voix des personnages et autres effets sonores liés aux combats par exemple.

Via le mode sample, il est possible d'appliquer des effets 3D au son : ainsi selon un repère tridimensionnel il est possible de définir la provenance de chacun des échantillons de sons joué. Bien que l'intérêt d'émettre du son depuis l'arrière du joueur soit limité dans un jeu de stratégie, le son est maintenant géré de cette manière, ce qui permet de jouer le son sur les haut-parleurs droits ou gauches, selon la position de l'unité émettrice du son dans le jeu par exemple.

Enfin, la gestion du matériel est entièrement prise en charge par FMod : si l'usager ne dispose pas de système sonore permettant de jouer le son physiquement derrière lui, le son sera transformé pour simuler des haut-parleurs arrière par exemple.

Le choix des musiques a été porté sur une piste de l'Album "The Wizard of Oz" de Francesco Lettera pour l'ambiance du jeu, et une de l'album "Alter Ego" de Cyril "Xcyril" Humbert. Les deux albums sont sous license Créative Commons, et nous disposons de plus de l'autorisation écrite des deux auteurs.

Les voix des différents unités ont quand à elles été enregistrées par nos soins.

Chapitre 5

Christopher : Moteur Graphique et Interface Utilisateur

5.1 Mon rôle dans le groupe

Au cours de ce projet je me suis chargé de deux parties importantes que sont le moteur graphique et l'interface utilisateur. J'ai également aidé à diverses tâches comme la communication et le réseau.

5.2 Enjeux et difficultés

5.2.1 Enjeux

Moteur Graphique

Le moteur graphique est vraiment la base du jeu. Cela concerne tout ce qui est affiché à l'écran. Aussi bien au niveau de l'affichage du terrain, des unités que de l'interface. Il est primordial qu'il soit flexible et optimisé.

Interface Utilisateur

L'interface utilisateur dans un jeu est primordiale, et qui plus est dans un jeu de stratégie. En effet, c'est l'unique moyen que l'utilisateur a d'interagir avec le jeu. Celle-ci doit être très riche afin de gérer le contrôle de plusieurs unités différentes.

Nous avons décidé d'utiliser les technologies Lua et XML, en oeuvre dans le jeu World of Warcraft pour exploiter ce qui se fait de mieux en matière d'interface. Cela permet de détacher la forme du fond. Cette séparation oblige

à se poser des questions sur la façon dont les choses vont être réalisées. Cela permet d'accélérer et de simplifier le développement.

5.2.2 Difficultés

J'ai passé une très grande partie de mon temps sur le chargement et l'affichage des modèles. En effet, le format des modèles est très peu documenté. Il m'a fallu découvrir par moi-même le fonctionnement par tâtonnement.

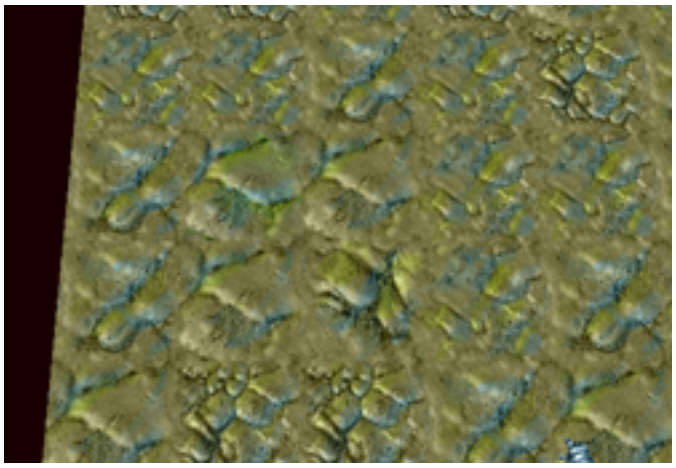
L'interface n'a pas été un problème étant donné mes connaissances. Ayant déja une idée très précise du travail à accomplir, il a suffit de le développer.

5.3 Récit de réalisation - Moteur Graphique

5.3.1 Affichage du terrain

La première chose à afficher lorsque l'on veut faire un Jeu de Stratégie reste la carte. Il s'agit en réalité d'un assemblage de carrés mis bout à bout avec sur chacun d'eux, une texture aléatoire. Certaines cases contiennent des textures spéciales, plus rares, pour diversifier le visuel. On peut par exemple penser à des cailloux ou a des fleurs.

L'affichage est modulaire, en effet, il permet d'afficher des textures dont le thème est différent. Si nous vient l'idée d'affichage une carte de montagne, ce sera parfaitement possible grâce à un sol rocheux.



5.3.2 Déplacement de la Caméra

Il est bien beau d'avoir une carte énorme mais encore faut-il pouvoir s'y déplacer. Après réflexion nous allons donner à l'utilisateur la possibilité de se déplacer selon 3 méthodes

- Les flèches directionnelles
- La souris sur le bord de l'écran
- Un « drag » de la souris sur la carte avec le bouton central appuyé

Ces trois méthodes ne sont pas en elles mêmes compliquées mais nécessitent des composants qui n'étaient pas encore été codés.

Tout d'abord il a fallu réaliser le module de gestion des entrées claviers. Celui-ci a tout de suite été pensé de façon à pouvoir changer les touches dans le futur.

Ensuite une gestion du curseur a été faite. Alors que nous étions en train de fonctionner dans un environnement en trois dimensions il a été nécessaire de comprendre comment passer dans le repère à deux dimensions que définit l'écran. Puis on place un carré texturé à partir d'une image contenant de nombreux curseurs à la position de la souris.

La où cela se complique c'est lorsque la souris est sur un bord de l'écran, celle-ci se transforme en un triangle indiquant la direction et le sens du déplacement de la caméra. L'image ne contient qu'une seule flèche donc il faut la tourner dans le bon sens. Ensuite la position d'application de la texture est différente selon chaque cas. La pointe de la flèche doit être à la position du curseur.

5.3.3 Affichage du modèle

L'étape suivante consiste à mettre de la vie sur cette terre vierge. Nous allons utiliser pour guise de personnages des modèles en format MDX utilisé par Blizzard dans Warcraft III.

Pour que le projet ne tombe pas sous le coup d'une utilisation illégale de matériel sous copyright, nous allons utiliser des modèles réalisées par des fans pour leurs cartes.

Les modèles en eux mêmes ne sont pas difficiles à charger, il suffit de traiter chaque bloc avec rigueur. Quant aux textures, il a fallu développer un petit gestionnaire afin de les gérer.

L'affichage statique se fait lui aussi sans aucun soucis, le format étant assez bien fait pour faciliter grandement tout le travail. Voici un Zerg réalisé par Abriko qui s'affiche dans notre jeu.

5.3.4 Optimisation

Frustum Culling

C'est après le constat désolant d'un jeu qui n'était plus fluide avec seulement deux unités affichées que nous nous sommes s'est intéressé plus en profondeur aux améliorations possibles. La première optimisation évidente est de ne pas afficher les unités si elles ne sont pas présentes dans le champs de vision. Alors que nous avions déjà un bout de code qui réalisait cette tâche de façon empirique, il a fallu faire appel aux mathématiques afin de trouver une solution.

Le problème est simple, on veut tester si un modèle est présent dans le champ de vision. Tout d'abord, il faut savoir que le champs de vision est constitué de 6 plans. On se retrouve donc à chercher si un modèle est inclut dans l'espace formé par ces 6 plans.

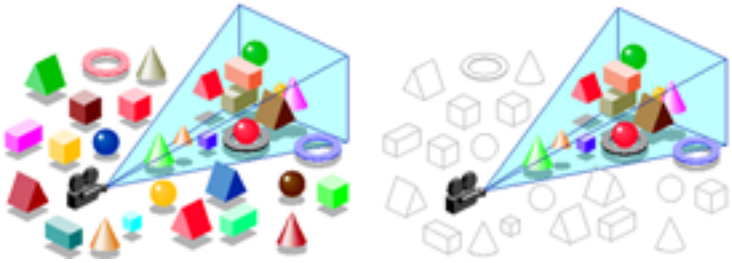


FIG. 5.1 – Elimination des modèles non visibles

La solution parfaite serait de déterminer si chaque triangle est visible ou non. Ainsi on afficherait uniquement les triangles visibles.

Le problème étant que le test de présence a un coût machine non négligeable par rapport à l'affichage d'un triangle. Donc on a pour idée de tester si une boîte contenant le modèle est présente dans le champs de vision. C'est à dire tester 8 points. Nous avons uniquement à disposition les coordonnées du diamètre d'une sphère qui contient le modèle entier. Après des recherches, il apparaît que le test de visibilité d'une sphère est très peu couteux en opérations comparés à celui d'un cube. Nous voici donc avec une élimination des modèles non visibles.

Vertex Array

La seconde grosse optimisation réside dans la façon d'afficher les polygones. Au départ nous utilisions la méthode traditionnelle d'OpenGL qui consiste à envoyer naïvement les coordonnées des vertex un par un à la carte graphique. Après de nombreuses recherches il existe plusieurs options pour accélérer le rendu. Celles-ci ont toutes pour principe d'envoyer les coordonnées des vertex groupées dans un tableau. C'est dans la disposition des données que divergent les méthodes.

Dans notre cas, nous allons d'abord envoyer l'adresse d'un tableau énorme contenant toutes les données affichables, puis donner un tableau plus petit contenant uniquement les indices des éléments à afficher. De cette façon la carte graphique va pouvoir récupérer les informations beaucoup plus rapidement.

Nous nous sommes également aperçu que pour le terrain et l'interface nous utilisions seulement deux coordonnées au lieu des trois. En effet, le terrain ainsi que l'interface n'ont pas de hauteur. Ainsi en retirant ce paramètre des éléments envoyés à la carte graphique nous avons pu obtenir des bénéfices notables en termes de performance.

5.3.5 Picking

Terrain

Le picking était déjà fonctionnel à la première soutenance mais ne concernait que les carrés de textures du terrain. Pour récupérer la position de la souris par rapport au terrain nous utilisons une autre méthode. Au lieu d'utiliser le picking nous utilisons encore des mathématiques. OpenGL met à disposition une fonction qui permet de récupérer les coordonnées de deux points de la droite projetée à partir de la souris vers le terrain. Il suffit ensuite de calculer l'intersection de cette droite et du plan d'équation z=0.

Models

Pour les modèles la technique du picking par OpenGL fonctionne très bien. Dans un souci d'optimisation nous effectuons également un test de champ de vision mais cette fois-ci sur un carré de dimensions 1x1, ou plus pour un rectangle de sélection.

5.3.6 Animations

Transformations géométriques

Il faut savoir que les animations sont en fait simplement des transformations géométriques effectuées sur les vertices du modèle. Grâce à seulement des translations, des homothéties et des rotations stockées sous forme de quaternions on arrive à obtenir toutes sortes d'animations.

Pour des soucis d'optimisation de l'espace les transformations liées aux animations ne sont prédéfinies que pour des « frames clés ». C'est-à-dire que par exemple si on gère une animation de quelqu'un qui lève le bras, on va prendre en photo le moment où il a le bras le long de son corps, puis en plein milieu de son mouvement et enfin en l'air.

Les frames intermédiaires sont calculées grâce à une simple interpolation linéaire en fonction du temps actuel et des deux frames adjacentes.

Parenté

Le modèle est subdivisé en parties de plus en plus petites qui héritent de l'animation de leur parent. Par exemple pour animer un corps on réalise l'animation du tronc par rapport au référentiel puis du bras par rapport au corps et enfin la main par rapport au bras.

Cela est en fait un système de modifications locales. En effet, il est inutile de devoir refaire l'animation des antennes d'un monstre si il court, il suffit de savoir la nouvelle orientation de la tête pendant le mouvement.

Affichage en contexte

Maintenant qu'il est possible de jouer l'ensemble des animations il faut savoir quand et comment les jouer. Nous avons répertorié trois familles d'animations.

- **Répétables** : Elles sont jouées en boucle. On peut par exemple citer la posture de repos de l'unité ou lorsqu'elle est en train de courir.
- **Une fois** : A la fin de l'animation le modèle reste sur la dernière frame. L'animation de mort est un exemple.
- **Ajustables** : La durée et l'avancement de l'animation sont déterminées par des facteurs extérieurs comme par exemple pour la construction de bâtiments.

Les animations répétables ont souvent plus d'une animation pour la même action afin de varier visuellement. Un indice de rareté est affecté à chacune d'elle afin d'en jouer certaines plus souvent que d'autres.

FIG. 5.2 – Animation de construction d'un bâtiment

5.3.7 Améliorations

Affichage

Nous nous sommes rendu compte que le système de parenté engendre le recalcul à de nombreuses reprises des transformations liées aux bornes principales. Un système de sauvegarde des transformations durant une frame a été mis en place. Nous avons noté une forte baisse du temps de calcul.

Une grande partie de l'affichage a été remanié afin d'en corriger les nombreuses erreurs. Elles n'étaient pas tant au niveau du code en lui même mais surtout des spécifications. En effet, dans la recherche de la frame clé lors des animations, c'est la frame juste avant le temps actuel qui est importante. Elle sert à faire le calcul de l'interpolation avec la frame qui suit. Il n'y a aucune prise en compte de la durée de l'animation comme j'avais cru au début.

Picking

Il était difficile de sélectionner des unités à cause de plusieurs facteurs. Tout d'abord certaines unités disposaient d'éléments énormes qui n'étaient que peu visible et occupaient beaucoup d'espace sélectionnable. Il s'est avéré qu'il existait un booléen disant si oui ou non la partie du modèle en cours pouvait être clickable. Nous l'avons donc incorporé.

Les bâtiments quant à eux ont une partie se situant en dessous du sol, cela les rendait donc sélectionnable en passant la souris sur le terrain juste en dessous d'eux. Afin de corriger ce problème nous ne prenons plus en compte ce qui se situe en dessous du niveau zéro.

Enfin lorsque plusieurs unités se chevauchent, il n'est plus possible de les sélectionner les deux en même temps par un simple clic. Seule l'unité la plus proche de nous sera sélectionnée. Par contre il suffit de faire un rectangle de sélection pour réactiver l'ancien fonctionnement.

5.4 Récit de réalisation - Interface Utilisateur

Dans un premier temps, nous avons affiché la partie graphique de l'interface réalisée par Unwirklich.

5.4.1 Fonctionnement

Structure

La partie XML contient tous les éléments de l'interface. Ils sont scindés en trois groupes.

Les frames sont les éléments essentiels de l'interface. Elles ne sont pas affichées mais permettent de donner une structure. Ce sont en quelque sorte des conteneurs, ou les plans de construction. Chaque frame possède en son sein les objets affichés mais également une série de sous-frames.

Les véritables éléments sont les textures et les FontStrings. Ils permettent comme leurs noms l'indiquent d'afficher soit des images soit du texte. On pourrait les apparenter aux pierres de l'édifice.

Relativité

Là où cette organisation nous fait gagner du temps c'est bien dans la relativité des éléments. En effet, toute partie est relative à une autre.

La relativité commence naturellement avec la relation Parent-Enfant qui s'effectue automatiquement grâce à la structure en profondeur du XML. Elle peut également être court-circuitée à n'importe quel endroit de la chaîne.

C'est dans son utilisation que réside son sens. Chaque élément est placé en fonction d'un autre. Dans l'image ci-dessous, la première icône a été placée

par rapport au bord inférieur droit de l'écran. Chaque icône qui suit est ensuite positionnée à quelques pixels à droite ou en bas de son voisin.

Une fois le premier élément placé il devient un jeu d'enfant de placer les autres. Les modifications spatiales deviennent grâce à ce système extrêmement rapide et intuitives.

FIG. 5.3 – Position relative des éléments par rapport au premier

Héritage

Il est possible de créer des frames qui ne seront pas affichées mais qui servent de modèle. Des frames, bien réelles cette fois-ci, vont être capables d'hériter de toutes les propriétés et objets du modèle. Ainsi afficher par exemple plusieurs boutons ayant un fonctionnement complexe, est l'affaire de quelques lignes.

L'héritage est une base pour l'élément, toutes les propriétés héritées peuvent être remplacées selon l'utilisation spécifique. Cela donne l'avantage d'établir des composants généraux qui peuvent être réutilisés facilement.

Tout d'abord, le système d'héritage a été largement revu afin de permettre d'enchaîner les templates. Par exemple on définit d'abord virtuellement un bouton puis l'on réalise une template avec ce même bouton ainsi qu'une image de fond pour obtenir le bouton final.

Interactivité

Afin de rendre cette interface interactive, nous avons fait le choix d'utiliser le langage de script Lua. Il est extrêmement simple à mettre en place et dispose d'une importante flexibilité qui permet de coder rapidement. N'étant pas compilé avec le programme, il est possible de prendre en compte les modifications sans avoir besoin de relancer le jeu.

Un système d'évènement a été mis en place afin de pouvoir appeler les fonctions programmées en Lua. On associe à chaque frame des évènements comme OnLoad, OnKeyDown ou encore OnMouseEnter. Il existe également OnEvent qui est appelé lors d'évènements relatifs au déroulement du jeu tel que l'arrivée d'un joueur.

Application : Barre de texte

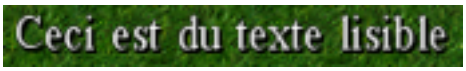
Afin de mettre en pratique le framework, nous avons dans un premier temps réalisé une barre de texte ainsi que l'affichage de messages. Maintenant que les bases sont posées, nous allons pouvoir nous consacrer au développement des modules de l'interface.

5.4.2 Texte Affiché

Nous utilisons GLFont[1] afin d'afficher du texte à l'écran. Cet outil est constitué de deux parties. Tout d'abord un exécutable qui permet à partir d'une police de notre choix de générer un fichier contenant une image de tous les caractères ainsi que des informations relatives à la taille et la position de chacun d'eux.

Une bibliothèque quant à elle permet de lire ce fichier et d'afficher du texte. Pour un meilleur contrôle, nous l'avons recodée. Celle-ci est maintenant partie intégrante du projet. Cela permet par exemple d'afficher du texte en plusieurs polices différentes.

La souplesse apportée a rendu possible des tests d'affichage en faisant varier des options afin d'obtenir un résultat lisible, mais également la possibilité d'afficher tous les caractères de la table ASCII, c'est-à-dire que les accents sont gérés.



Les textes sont gérés comme les textures et les frames au niveau de l'interface, c'est-à-dire qu'ils peuvent être placés relativement ou servir de template. Aussi nous avons ajouté la possibilité de changer la couleur de celui-ci.

FIG. 5.4 – Position relative du texte et couleur

Pour aller encore plus loin dans l'affichage du texte il fallait pouvoir changer la taille de la police. L'utilisation de GLFonts pour afficher le texte nous oblige à avoir un fichier de police différent par taille. Changer la taille revenait donc à pouvoir afficher plusieurs polices.

D'un point de vue technique, charger plusieurs polices n'a pas été compliqué, mais les textures générées sont rentrées en conflit avec les vraies textures du jeu. Il a donc fallu coder une procédure d'appel générique aux textures qui s'est au final révélée beaucoup plus performante que l'ancienne.

[1]GLFont, http ://students.cs.byu.edu/ bfish/glfont.php

5.4.3 Cadres

Pour réaliser le menu, il a fallu séparer l'espace en plusieurs parties. Le moyen largement utilisé est la création de cadres. Or pour ce faire il est nécessaire d'avoir des bordures. Aussitôt dit, aussitôt fait. A partir d'une texture contenant les bordures on obtient un beau cadre.

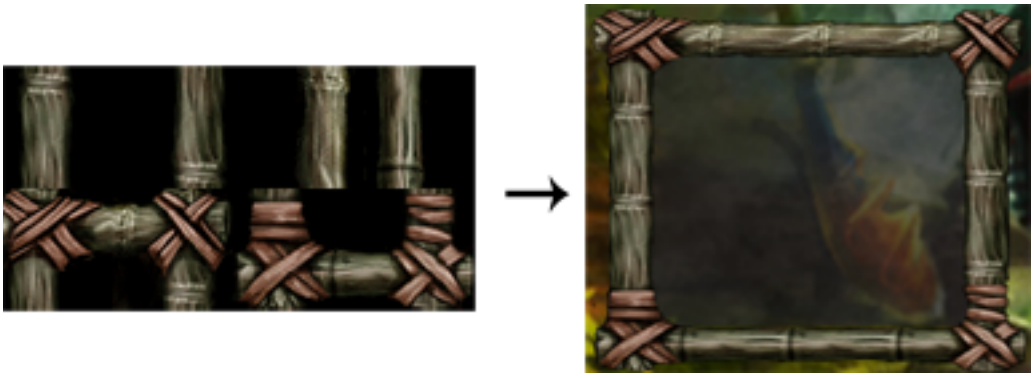


FIG. 5.5 – Affichage d'un cadre à partir d'une texture

5.4.4 Contrôles

Toujours dans le cadre du menu, l'interaction avec le joueur doit se faire grâce à des contrôles tels que des boutons, des champs de textes ou encore une liste d'éléments. Grâce à la modularité de l'interface il fut très simple de créer des objets virtuels dotés de scripts qui puissent être réutilisé et modifiés à volonté.

Boutons

Les boutons sont relativement basiques. Ils réagissent à la souris de deux façons : passer la souris au dessus met en surbrillance le bouton alors que cliquer dessus l'enfonce. L'action du bouton ne sera effectuée que si la souris est relâchée au dessus du bouton. Il existe également un état désactivé.

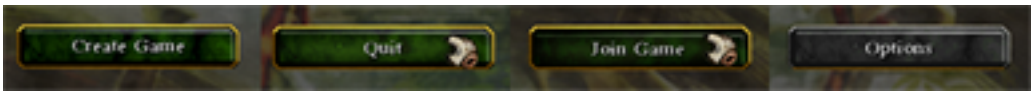


FIG. 5.6 – Différents états des boutons

Liste

Dans le but d'afficher des informations telles que la liste des serveurs ou la liste des personnes connectées il était nécessaire de coder une liste générique qui puisse les présenter de façon lisible et compacte. Pour interagir avec celle-ci, une sélection a été mise en place qui peut être effectuée grâce à la souris mais aussi avec les flèches haut et bas.

Vous pouvez remarquer que tout se réutilise, en effet la bordure du champ est la même que la bordure du cadre mais avec une taille moindre. N'étant pas nous même artistes, il faut savoir exploiter au maximum toutes les ressources disponibles.

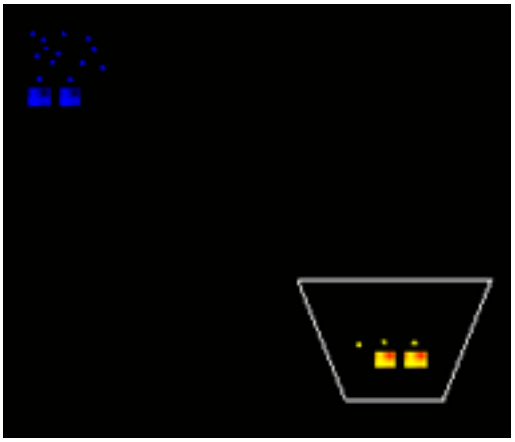
Champs de texte

Pour laisser l'utilisateur saisir des informations il lui faut un champ de texte. Celui développé, outre le fait d'ajouter les caractères au texte au fur et à mesure qu'ils sont tapés, permet de se déplacer à l'intérieur via les touches gauches et droites. Il détecte également la combinaison CTRL + V afin de copier le texte dans le presse papier.

Lorsque deux champs de texte sont affichés, afin de ne pas écrire dans les deux à la fois nous avons mis en place un petit système de « focus ». Il faut pour pouvoir écrire du texte dans un champ le choisir en cliquant dessus. Il devient alors responsable des informations entrées au clavier.

5.4.5 Minicarte

On s'est rendu compte que la minicarte était un élément indispensable au jeu, elle permet de se repérer dans la carte. Elle affiche à l'échelle l'ensemble des unités présentes. Les bâtiments sont représentés par des carrés tandis que les unités par des cercles. Les formes sont colorées en fonction du joueur qui contrôle les unités.



5.5 Mes impressions

5.5.1 Moteur Graphique

Le travail au niveau des modèles fut fort instructif. J'ai découvert une application mathématique de beaucoup de concepts étudiés au lycée, avec toute la géométrie notamment, mais aussi de nouveaux outils comme les matrices et les quaternions.

J'ai été obligé de faire un gros effort sur l'optimisation, l'affichage et le calcul des transformations demande énormément de ressources. Au départ afficher deux modèles statiques relevait du défi même pour une machine performante. Alors que maintenant le jeu reste fluide avec une vingtaine de modèles à l'écran.

Je serai intéressé de savoir quelles autres techniques les développeurs de Warcraft ont utilisé pour leur moteur, car je suis encore très loin de leurs performances.

5.5.2 Interface Utilisateur

Mon rôle dans l'interface a certes été de la concevoir techniquement mais une part importante fut également l'apprentissage de cette nouvelle façon de développer à Alban et Félix. Il a fallu faire des exemples simples d'utilisation qui puissent leur permettre de démarrer mais également rester auprès d'eux afin de corriger leurs erreurs et leur inculquer une méthode de développement générale.

De plus, le fait qu'ils programment sur l'interface a fait apparaître de nombreux bugs et possibilités manquantes que j'ai du réparer et combler. Au final notre collaboration apparaît comme réussie au vu des résultats obtenus.

5.5.3 Travaux Annexes

Je ne me suis pas cantonné au moteur graphique et à l'interface, je me suis également occupé de tout un tas de choses dans le projet. J'ai été à la base de la réunification des différentes parties de chacun à la suite de la première soutenance ainsi que de la structure physique des fichiers.

Je me suis fortement rapproché d'Alban dans toutes ses tâches. Je l'ai aidé grâce à mes connaissances en développement web pour le site mais j'ai aussi activement collaboré au niveau de la communication. Tout d'abord avec EpiPub pour lequel j'ai participé aux débats lors du développement. Je suis à l'origine du logo et de la vidéo de présentation et je l'ai aidé à promouvoir le projet.

Nous avons également monté le projet de stand EPITA au Festival du Jeu Vidéo où nous allons présenter notre jeu ainsi que les meilleurs projets de cette année. Il a consisté en la rédaction d'un rapport d'une quinzaine de pages ainsi qu'à convaincre l'administration de l'intérêt de l'évènement.

Enfin j'ai mené les discussions concernant le thème du jeu : les différents personnages, sorts, différents types de récolte de ressources. Après décision j'ai été chargé de l'intégrer au jeu.

5.5.4 Conclusion

Je suis très fier d'avoir participé à ce projet. J'ai durant de nombreuses années été très actif dans la modification des jeux de Blizzard autant par l'éditeur de cartes de Warcraft III que les addons de World of Warcraft. C'est donc pour moi une façon de passer de l'autre côté du décor, de me mettre à la place de vrais développeurs.

D'autant que nous avons réussi à former une équipe comportant des personnes très différentes mais toutes compétentes dans leur domaine, et réussi à coordonner nos efforts pour parvenir à un très bon résultat.

Chapitre 6

Vladimir : Déplacement des Unités

6.1 Mon rôle dans l'équipe

Une de mes fonctions dans le groupe est celle de chef en plus d'être le responsable du déplacement des unités. En effet, nous savions que le projet serait assez conséquent, il était donc important de définir des objectifs plutôts clairs pour rendre possible une vision globale de l'avancement du jeu. Bien que nous ayons décidé de la nature du projet ensemble, et bien que chaque décision ait toujours été prise en fonction des avis de tout le monde, il était important qu'une personne tranche dans certains cas. Il a donc toujours fallu que je garde en tête les objectifs principaux de notre projet, pour essayer de ne jamais passer à côté d'un aspect prévu à la base, ou encore que je réfléchisse précisément à ce que je voulais que le jeu fasse. Le fait que l'un d'entre nous puisse trancher a permis notamment de ne pas s'attarder sur certaines hésitations. Je me suis également tout le long du projet assuré que chacun avait sa part de travail, qu'elle correspondait aux attentes et au cahier des charges. Enfin, principalement grâce à mon expérience en programmation, j'ai pu aider certains membres du groupe à définir la structure du travail qu'on attendait d'eux. Je me suis donc documenté sur la plupart des parties de notre projet, principalement la gestion du réseau, en plus de ma partie.

D'autre part, mon rôle principal était de m'occucper du déplacement des unités, abordé dans la partie suivante.

6.2 Déplacer des unités

Le déplacement d'unités dans un jeu de stratégie est généralement une des parties les plus importantes, puisqu'ici il ne s'agit pas simplement de déplacer une unité linéairement entre 2 points selon une vitesse, le joueur de RTS attend généralement beaucoup plus de la part de ses unités. En effet, il s'agit d'un jeu de stratégie et non d'adresse ; dans la mesure où un joueur doit parfois diriger plusieurs dizaines d'unités à la fois, il serait assez malvenu de laisser le joueur déplacer chaque unité indviduellement et précisément. Le joueur préfèrera bien entendu former des groupes de plusieurs unités, leur donner un objectif simple, comme de se rendre à un endroit ou d'effectuer une patrouille, et en attendant s'occuper d'autres unités, de son économie, etc. Il attend d'autre part que ses unités trouvent leur chemin seules, c'est à dire qu'elles s'arrangeront pour contourner les obstacles sur leur chemin, et essairont d'emprunter un des chemins les plus courts pour rejoindre leur destination. Les phases de batailles sont généralement fréquentes dans ce type de jeu, il est donc intéressant de prévoir des attitudes d'unités utiles pendant ces situations, telles que des systèmes d'attaque automatique, permettant à une unité inactive sur un champ de bataille qui subirait des dégats de riposter automatiquement, ou encore pour un archer d'attaquer un ennemi à portée. Un système de formation est également intéressant, permettant aux unités faibles comme des archers de rester derrière des fantassins qui ont plus leur place en première ligne.

6.3 Enjeux et difficultés

La gestion de déplacement des unités doit soulager le joueur de tâches fastidieuses comme diriger précisément ses unités unes à unes pour leur faire éviter des obstacles, s'occuper de la riposte de chaque unité attaquée, lorsqu'une unité en attaque une autre, si son adversaire décide subitement de se déplacer, il serait intéressant que notre unité suive son adversaire.

Techniquement, plusieurs aspects interviennent dans ces attitudes que doivent adopter les unités. Il va tout d'abord y avoir un système d'ordre aux unités, qui permettra de définir des attitudes spécifiques, comme se déplacer, attaquer, patrouiller, construire, etc. Chaque unité possède également une attitude face aux ennemis : un guerrier aura tendance à attaquer les ennemis à portée tandis qu'un constructeur sera pacifique. Les unités doivent aussi fonctionner par groupe : un ordre donné à un groupe d'unités doit se répercuter correctement sur chaque unité. Il est par exemple inutile de demander à chaque unité d'un groupe de se déplacer exactement même endroit

puisque 2 unités ne doivent pas pouvoir se chevaucher, il est plus judicieux de s'assurer qu'elles vont chacune trouver un endroit adequat pour aller. Il est également important qu'une unité qui doit se déplacer entre deux points évite les obstacles d'elle même. Pour cela, des algorithmes de recherche de chemin le plus court sont nécessaire. Enfin tout ce système doit paraître le plus intuitif et transparent possible à l'utilisateur. C'est à dire que demander un déplacement en évitant des obstacles ne doit pas ralentir le jeu, et il ne faudrait pas qu'une unité semble prendre un chemin aberrant contournant par exemple tout un village quand le traverser est plus rapide.

6.4 Première soutenance

Je me suis d'abord concentré sur ce qui m'a paru le plus urgent, à savoir comprendre et implémenter un algorithme de recherche de chemin le plus court (plus communément appelé algorithme de pathfinding).

En effet, c'est avec un algorithme de ce type que les unités pourront non seulement chercher un chemin mais aussi éventuellement si elles rencontrent des obstacles imprévus, recalculer leur chemin. Il m'a donc semblé important de pouvoir produire une fonction générique de recherche de chemin.

Après quelques recherches sur le sujet, mon choix s'est rapidement porté sur l'algorithme de l'A* (A star pour les informaticiens, A étoile pour les professeurs d'algorithmique), qui m'a semblé suffisamment simple au début et qui après plusieurs optimisations semble être un des algorithmes de path-finding basiques les plus satisfaisant au niveau des performances.

Globalement, cet algorithme permet de trouver un chemin la plupart du temps parmi les meilleurs chemin possibles, mais par rapport à d'autres algorithmes qui trouvent toujours le meilleur chemin, il est très rapide. Une explication de l'algorithme A* est donnée en annexe.

A ce stade, seul l'algorithme de plus court chemin était opérationnel, et comme nous n'avions pas encore de jeu à proprement parler, il fonctionnait dans un exécutable avec sa propre interface graphique afin de visualiser le comportement et les corrections éventuelles nécessaires.

6.5 Deuxième soutenance

A cette soutenance, le mot d'ordre était d'intégrer nos travaux dans un seul exécutable. Cette étape très importante n'était néanmoins pas évidente, mais nous a permis par la suite de concrètement tous travailler sur le même projet. C'est à cette soutenance que nous avons commencé à coordonner nos

travaux grâce au système de sous-versions SVN.

6.5.1 Recherche de chemin

Plusieurs améliorations ont été apportées à l'algorithme de pathfinding à cette soutenance. Notament ajouter un moyen d'obtenir un chemin vers le point le plus proche de la destination demandée dans le cas où le point d'arrivée ne serait pas accessible. Cette amélioration permet au joueur de ne pas réfléchir exactement à la position d'arrivée souhaitée : il peut donner une position imprécise, et si elle s'avère impossible à atteindre, l'unité effectuera tout de même un mouvement vers cet endroit plutôt que de ne pas bouger. Un autre point important est que dans un monde représenté par un échiquier, un chemin rectiligne qui traverse par exemple 5 cases, se fait en 5 étapes, alors qu'il suffirait de demander une seule étape avec comme destination la fin de la ligne droite.

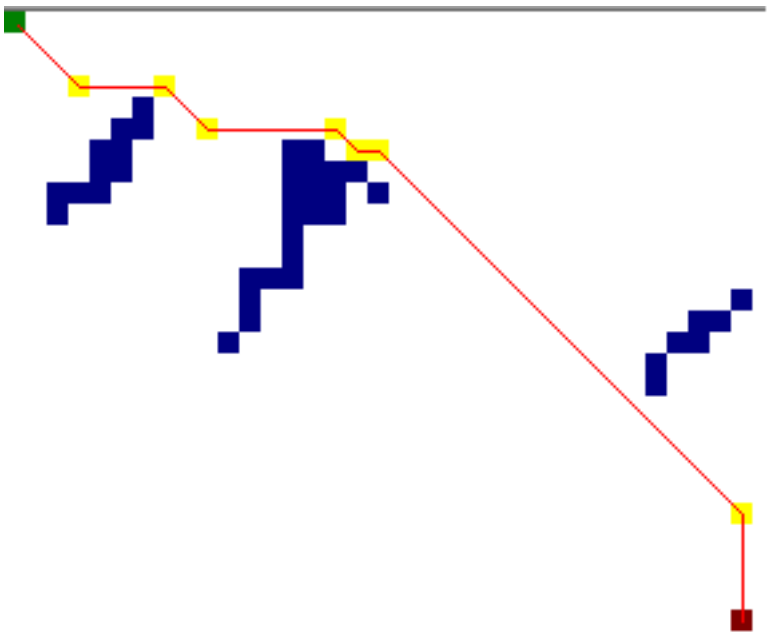


FIG. 6.1 – Au lieu de lister toutes les cases du chemin, l'algorithme ne donne que des points pivots (en jaune)

Des optimisations nécessitant des notions algorithmiques plus avancées que celles que j'avais à ce stade ont également été apportées à l'algorithme pour le rendre plus rapide. Enfin, dans un jeu tel que le nôtre, les unités et même les obstacles n'ont pas tous la même taille qui pourrait servir par

exemple de taille standard de case. Il faut donc prévoir que l'algorithme ne cherche pas forcément un chemin prévu pour une case, mais qu'il évite par exemple de passer entre 2 obstacles trop rapprochés.

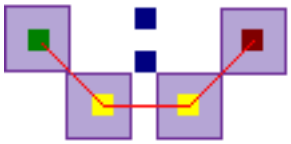


FIG. 6.2 – Ici, l'unité ne peut pas passer entre les 2 obstacles, l'algorithme les contourne donc

Pour cette soutenance, la recherche de chemin était donc intégrée au jeu, plus intelligente et aussi plus rapide.

6.5.2 Système d'ordres

En plus de leur trouver un chemin, mon rôle consiste à déplacer les unités, et particulièrement à gérer un groupe d'unités. En étroite collaboration avec le responsable de gestion des données et le responsable moteur 3D et interface, nous avons mis en place un système permettant de sélectionner les unités à l'écran, de gérer cette selection, et de leur donner des ordres. Le système de gestion d'ordres à une unité a permi de facilement passer d'un mouvement rectiligne sans tenir compte des obstacles, à un appel à la fonction de recherche de chemin.

Ce système permettait lors de la seconde soutenance de :

- sélectionner une ou plusieurs unités alliées.
- sélectionner une unité ennemie.
- donner l'ordre d'attaquer un ennemi
- donner l'ordre de se déplacer
- donner l'ordre de construire ou de réparer un bâtiment
- donner l'ordre de suivre un allié

6.6 Troisième soutenance

6.6.1 Recherche de chemin

Un des plus gros problèmes du pathfinding lors de la deuxième soutenance était que dès qu'une recherche de chemin ne pouvait aboutir, l'algorithme parcourait toute la carte à la recherche d'un chemin éventuellement

correct. Bien que le seul moyen de vérifier qu'il n'existe aucun chemin vers une destination est de tous les emprunter, il est possible de réduire les gros ralentissements causés par cette recherche négative.

Le premier point a été de chercher des optimisations basiques dans l'algorithme initial sans y apporter de modification. En effet, il s'est averé que l'intégration de l'algorithme dans le jeu[1] d'une manière un peu trop naïve n'était pas du tout optimisée, et après un peu de recherche il a été possible de réduire le nombre de calculs de manière drastique.

Le deuxième problème le plus évident était que de nombreuses recherches négatives pouvaient être évitées. En effet, lorsque l'on demande à une unité d'aller vers un bâtiment, par exemple pour le réparer, on sait pertinemment que l'unité ne pourra pas arriver à la position du bâtiment sans rentrer en collision avec ce dernier. L'algorithme de recherche de chemin prenait donc aussi en compte la portée de l'unité qui devait se déplacer : Un chemin était désormais satisfaisant dès qu'il permettait à l'unité d'être à portée de sa cible et non plus lorsqu'il menait à la cible directement. Un archer cherchant à tirer sur un ennemi hors de portée aura donc un chemin menant suffisament près plutôt qu'un chemin menant à l'ennemi lui-même.

Enfin le système ne prennait à partir de ce moment plus en compte les unités qui se déplacaient, puisqu'elles étaient susceptibles de ne plus être à la même position pendant le déplacement.

6.6.2 Ordres et mouvements

Les unités pouvaient à partir de cette soutenance adopter une attitude agressive, ou pacifique. Ainsi une unité agressive aura tendance à attaquer les unités ennemie dans son champ de vision tandis qu'une unité passive ne s'en occupera pas. Evidemment le comportement agressif est idéal pour les unités de combat, ou les structures défensives, tandis que l'attitude passive correspond à l'attitude attendue d'unités de type constructeurs ou récolteurs. L'agressivité concerne aussi les ordres. Un ordre de déplacement peut ainsi être passif, utile pour fuire, ou agressif, pour que l'unité attaque tous les ennemis rencontrés en chemin et dans son champ de vision.

Un nouvel ordre a été ajouté à cette soutenance : l'ordre de patrouille. Les unités étaient désormais capables de patrouiller entre plusieurs points sur la carte. Une unité en patrouille fait tout simplement des aller-retours entre tous ses points de patrouillle, et comme le but est de surveiller les activités ennemies, cet ordre est de type agressif. Cet ordre est utile pour surveiller

[1] l'algorithme lui même ayant d'abord été réalisé et testé dans un exécutable indépendant

les alentours d'une zone, ou encore pour balayer la carte au peigne fin avec un groupe pour trouver et détruire les derniers survivants adverses.

6.7 Soutenance Finale

Cette soutenance étant l'étape finale du projet, il a fallu réfléchir à éliminer certains vecteurs de recherche, et se concentrer sur la finition de ce que nous avions déjà. Il était cependant hors de question de ne pas ajouter de nouvelles fonctionnalités. Ainsi, en plus de quelques optimisations apportées à l'algorithme de recherche de chemin, toujours pour éviter des ralentissements pendant le jeu, plusieurs nouveautés ont fait leur apparition au niveau du déplacement des unités. Ainsi, un système de groupe existe désormais dans le jeu. Ce système permet de répartir des ordres entre plusieurs unités à la fois. Il était évidemment possible de donner un ordre à plusieurs unités à la fois avant, mais ceux-ci étaient simplement répétés individuellement à chaque unité, la notion de groupe était donc inexistante. Le nouveau système de groupe permet de répercuter chaque ordre de manière pertinente sur les membres qui le composent. Ainsi lorsque l'on demande aux unités de s'arrêter, on peut simplement donner l'ordre à toutes les unités, autant lors d'un déplacement il est intéressant de ne pas toutes leur donner la même destination.

Un système de formations est également apparu : les unités qui se déplacent peuvent adopter des formations prédéfinies, comme par exemple une formation carrée. Les formations permettent aussi de faire évoluer ses groupes d'une manière stratégique sur les champs de bataille. En effet, un archer n'a pas plus sa place en première ligne qu'un fantassin à l'arrière. Il reste évidement le choix au joueur de détailler ses déplacements en prenant chaque unité, mais des déplacements minutieux pourraient lui faire perdre du temps pour gérer son économie ou ses constructions.

6.8 Mes impressions

J'ai vraiment apprécié ce projet tout au long de sa durée. Il a été très enrichissant pour moi car j'ai dû me documenter sur de nombreux sujets qui m'étaient inconnus, ou approfondir mes connaissances dans certains domaines. J'ai énormément appris que ca ne soit d'un point de vu algorithmique, la gestion d'un projet de cette envergure, et le travail en groupe. Je n'avais en effet jamais eu la possibilité de réaliser un projet informatique en groupe, et cette expérience s'est avérée très différente de la création de programme seul.

Au lieu de ça, le travail en groupe change beaucoup d'aspect. Il n'y a pour ainsi dire jamais eu de problème dans le groupe, que ca soit dans l'entente ou dans le travail de chacun.

Chapitre 7

Félix : Gestion des données et interface

7.1 Mon rôle dans le groupe

Je me suis occupé de toute la gestion des données du jeu en temps réel mais aussi du stockage de ces données en externe. Dans un jeu de stratégie classique, tout l'intérêt du jeu repose sur la possibilité de choisir le développement technologique et militaire que l'on préfère tout en prévoyant les éventuelles manoeuvres de l'adversaire afin d'adapter sa stratégie. Un autre aspect important est la gestion des actions de ces unités avec beaucoup de souplesse afin de permettre un rythme de jeu rapide.

7.2 Présentation des mécanismes du jeu

Les unités se doivent d'être équilibrées et variées. Tout d'abord l'unité de base est un simple constructeur, peu résistant et possédant une faible force de frappe, son rôle sera de développer son camp en créant des bâtiments plus avancés qui eux permettront la formation d'une vaste et puissante armée.

Les unités plus évoluées sont ensuite crées via des bâtiments, parmi ces unités, il y a plusieurs types.

La première unité possible est une unité basique de combat rapproché, très résistante, capable de gros dommages au corps à corps, mais le plus difficile est peut être d'arriver à portée d'attaque.

Les unités de tir à distance forment un second type d'unités, elles attaquent de loin, sont capables d'infliger des dommages significatifs mais sont très vulnérables une fois engagées en combat rapproché.

Enfin les lanceurs de sorts sont généralement peu résistants mais peuvent posséder des sorts de dégâts directs dévastateurs ou encore lancer des améliorations sur eux ou leurs alliés avant d'augmenter leur capacité d'attaque ou de défense.

Pour faciliter la gestion d'unités, des groupes d'unités peuvent être mémorisés puis rappelés par le biais de raccourcis claviers. Chaque unité possède toute une série d'ordres de bases en plus de ses éventuels sorts et compétences, ces ordres sont : se déplacer, attaquer, tenir la position, s'arrêter, patrouiller et réparer si c'est un constructeur.

7.3 Enjeux et difficultés

L'objectif final est de permettre des parties équilibrées et rythmées entre joueurs lors de parties en ligne.

7.3.1 Enjeux

Concernant l'équilibre du jeu, aucune unité ne doit être surpuissante, chaque unité doit posséder ses forces et ses faiblesses, c'est-à-dire qu'elle doit être la plus efficace dans une certaine situation ou associée avec une autre unité, mais elle doit être très vulnérable dans la situation de prédilection d'une autre unité.

Pour créer un rythme de jeu, il est nécessaire de proposer des stratégies variées aux joueurs, les armées s'opposant ne doivent pas être des clones, mais elles doivent refléter les choix stratégiques effectuées par le joueur en début de partie. Ainsi les unités doivent posséder de nombreuses compétences spéciales pour créer autant de stratégies différentes.

Une grande importance doit être accordée au fait de lancer la bonne action au bon moment, par exemple lancer un sort de dégâts directs afin d'achever une unité avant que le joueur adversaire ne puisse la soigner. De cette manière, le joueur doit avoir accès à toutes les informations dont il a besoin pour prendre des décisions rapidement, l'interface se doit donc d'être intuitive et précise et les ordres des unités doivent pouvoir être réalisés par des raccourcis clavier et non pas seulement a la souris. De plus, la manipulation de groupes d'unités doit être facilitée pour les batailles de grande envergure.

7.3.2 Difficultés

Les difficultés consistent à ordonner et à optimiser tout ces aspects. L'interface doit d'abord être intuitive, c'est-à-dire qu'elle doit présenter des icones explicites mais aussi afficher des info-bulles pour donner des informations au joueur concernant une compétence ou un ordre. Elle doit également être optimisée, il ne s'agit de limiter les interactions entre Delphi et Lua, par exemple les données ne doivent pas toutes êtres mises à jour en permanence mais seulement quand un événement spécial se déroule.

Pour la gestion des données en temps réel, l'optimisation est aussi de rigueur tout en veillant au bon déroulement des déplacements, actions des unités et effets temporels des sorts. Une partie importante est de collaborer avec les autres membres de l'équipe afin d'adapter pour répondre aux exigences et impératifs provenant de la recherche du plus court chemin d'une part et de la nécessité d'une synchronisation parfaite de ces données par le réseau.

7.4 Récit de réalisation

7.4.1 Première Soutenance

Réflexion

Ma première étape a été la décision d'un modèle de gestion de données préalablement à tout développement. Ce travail a été effectué lors du temps écoulé entre la formation du groupe et la première soutenance du projet, je ne possédais aucune expérience de programmation jusqu'à mon entrée a l'Epita, et le Delphi s'est avéré très différent du Caml de début d'année, une grande partie du temps s'est vu donc consacré à l'apprentissage de ce nouveau langage impératif à travers la création de mini programmes et d'expérimentations pour le projet tout cela avec l'assistance et les conseils de Christopher.

Une analyse de ce que j'avais à réaliser a rapidement montré que je devrais stocker beaucoup d'informations en dehors du jeu, le choix du format XML s'est donc rapidement amorcé afin d'avoir une base de données d'unités et de compétences claire, lisible et facilement modifiable. Un fichier XML est également dédié aux préférences de l'utilisateur concernant les options (sons, résolution, mode fenêtré, etc.). Ainsi le premier exécutable créé fut un utilitaire de modification des options du jeux tel que le volume sonore, la résolution, le choix du mode large ou fenêtré.

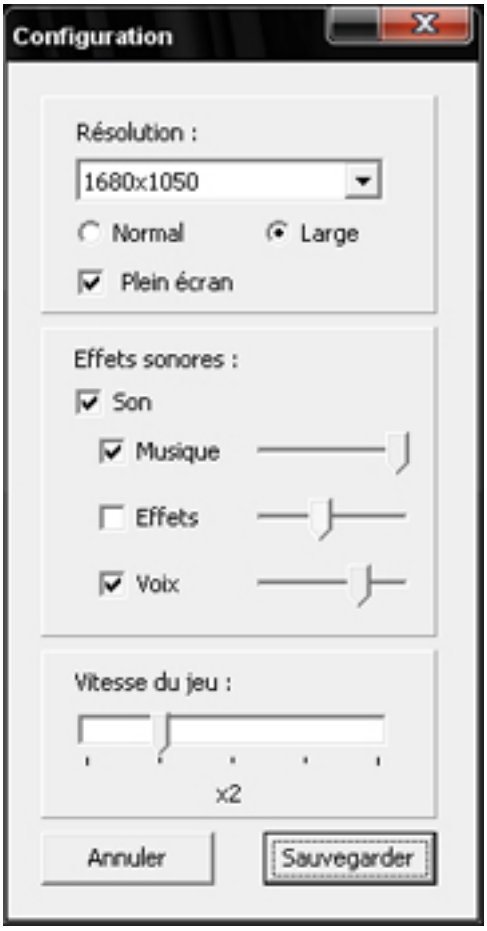


FIG. 7.1 – Utilitaire de modification d'options destiné a l'utilisateur.

Commencement

Apres avoir réfléchi à un paradigme de gestion de données, j'ai décidé de commencer son développement et de le mettre à l'épreuve en créant une simulation de gestion par le biais d'une TForm, les TForm présentent l'avantage d'être adaptées à ce genre de test, dans mon cas, cela m'a permis d'avoir visuellement toutes les informations du jeu pour vérifier que tout se déroulait parfaitement. Ainsi il était possible de créer une des unités chargée depuis le XML et de regarder ses caractéristiques, puis de lancer éventuellement un sort afin d'en voir les effets. Le temps était pris en compte puisque les sorts avaient une durée limitée. Les unités pouvaient également se déplacer virtuellement.

Cette première réelle expérience m'a donné un bon aperçu de ce que j'aurais à réaliser, je me suis rendu compte des différentes difficultés pouvant survenir et j'étais surtout prêt à intégrer le résultat de mon travail au moteur du jeu réalisé par Christopher.

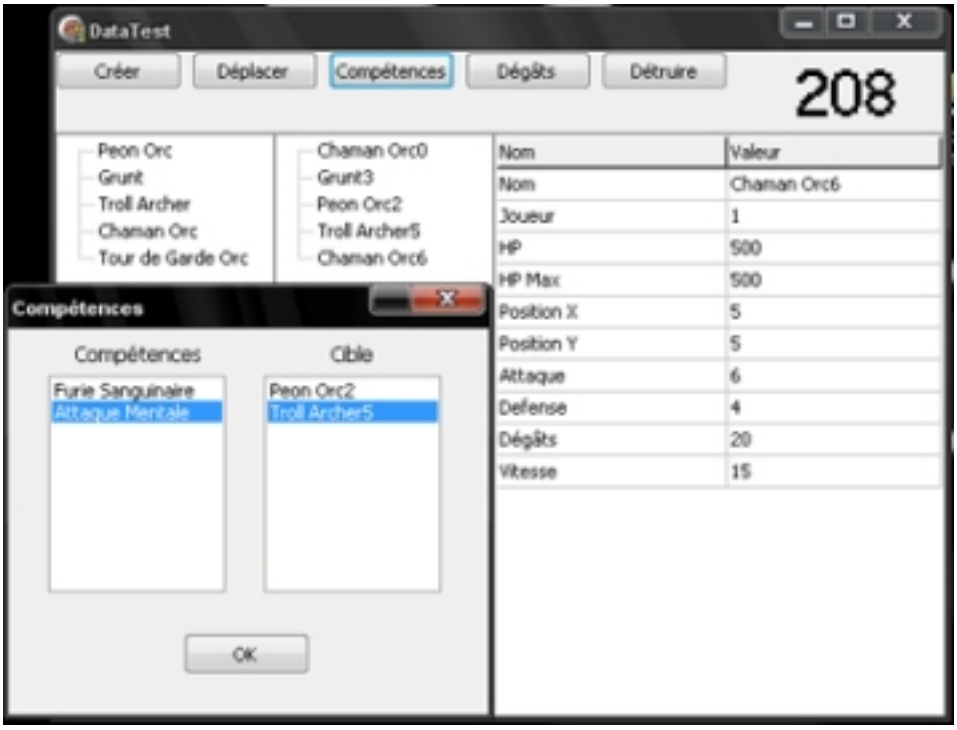


FIG. 7.2 – Simulation de gestion des données en temps réel.

7.4.2 Deuxième Soutenance

Intégration et interactions

J'étais conscient du fait que mon modèle de gestion allait devoir être confronté à la réalité du jeu, c'est-à-dire que d'autres problèmes allaient surement survenir, la convergence vers le travail de Christopher s'est promptement amorcée dés la reprise du travail après la première soutenance.

La finalité permanente recherchée lors de notre travail coopératif a été de développer une interaction intuitive et performante entre le jeu et l'utilisateur. Mon but premier était d'intégrer toutes les possibilités offertes par la gestion de données en TForm de la soutenance précédente tout en surmontant les éventuelles difficultés rencontrées lors de l'adaptation de mon modèle de gestion abstrait à un nouveau mode de fonctionnement plus visuel où la base de l'interaction pour le joueur était devenu la sélection de une ou plusieurs unités.

La création et la sélection d'une unité à d'abord été rendu possible grâce au picking réalisé par Christopher, ensuite ; afin de mener à bien la suite de mes tests, j'ai affiché les caractéristiques des unités de manière simple sur l'interface. La prochaine étape atteinte fut l'ajout de la gestion des compétences et de leurs effets tout en prenant en compte leur durée.

FIG. 7.3 – Cercle de sélection

La création de bâtiments a été ajoutée avec une vérification de la place disponible sous le curseur grâce à un carré de validité coloré. Dans l'exemple ci-dessous, l'unité sélectionnée ne peut pas construire à l'emplacement désiré car le bord inférieur gauche est déjà occupé par un autre bâtiment.

FIG. 7.4 – Impossibilité de construire sur un bâtiment

Afin de permettre au joueur de contrôler plus facilement des groupes d'unité, j'ai également ajouté la multi sélection et la gestion de groupes mémorisables à l'aide des raccourcis claviers.

Début de convergence

Sur un plan collectif, cette soutenance a marquée la convergence définitive avec le moteur graphique de Christopher, un travail important s'est aussi déroulé en symbiose avec Vladimir afin d'adapter nos parties respectives dans l'optique d'implanter son algorithme A Star de recherche de plus court chemin dans le jeu.

Cette coopération a mise en lumière la nécessité de créer un système d'ordres le plus générique possible afin de pouvoir en créer de nouveaux et pour gérer leur résolution de manière analogue. Nous avons donc décidé de créer une file d'ordre pour chaque unité, ainsi plusieurs ordres peuvent être exécutés à la suite sans nouvelle intervention du joueur.

Concernant mon travail en duo avec Alban, nous avions réussi à créer des unités par le réseau et à leur infliger des dégâts.

De nombreuses choses restaient encore à faire comme la poursuite de la convergence avec le travail de Vladimir et un important travail sur la synchronisation et l'envoi de données par le réseau.

7.4.3 Troisième Soutenance

Compétences et effets

J'ai d'abord travaillé sur les compétences et leurs effets. Il a été possible de rajouter un effet périodique à ces compétences en précisant un intervalle d'effet. Les compétences pouvaient aussi avoir désormais un effet de zone, une durée infinie, ou encore d'être active des la création de l'unité sans lancement préalable. Afin de pouvoir créer des compétences avec des effets spéciaux non traductibles par quelques champs dans un fichier XML, les compétences pouvaient aussi appeler une procédure Delphi lors de leur exécution afin de pouvoir déclencher cet effet.

Ensuite, j'ai implanté une file de construction d'unités pour les bâtiments, les unités sont crées sur la position disponible la plus proche.

Lors des précédentes soutenances, les unités à distance ne pouvaient pas attaquer, j'ai travaillé dans ce sens, elles pouvaient alors attaquer si elles étaient a portée et un modèle de projectile apparaissait et se dirigeait vers la cible Les projectiles pouvaient avoir soit une trajectoire linéaire comme par exemple pour une flèche ou un carreau d'arbalète, ou alors une trajectoire parabolique pour les projectiles de type boulets de canon.

Interface

La deuxième partie de mon travail s'est surtout axée sur le développement de l'interface grâce aux interactions entre Delphi, Lua et XML afin de la rendre agréable visuellement mais aussi intuitive et complète. A la sélection d'une unité, les ordres et compétences disponibles s'affichent dans le cadre en bas à gauche sous formes de boutons cliquables avec une icône explicite. Cependant, au lieu de cliquer sur ces boutons, le joueur peut choisir de lancer des compétences ou des ordres grâce aux raccourcis claviers prédéfinis.

Des fonctionnalités ont été ajoutées pour que le joueur puisse accéder aux informations concernant les unités, tout d'abord en bas de l'écran apparaît toutes les caractéristiques de l'unité sélectionnée tel que le nom, l'attaque, la défense, les points de vie et points de mana mais aussi les effets actuellement actifs sur la cible. De plus, une barre de vie s'affiche au dessus du modèle de l'unité lorsque l'on passe le curseur par-dessus ou si on appuie sur la touche ALT.

FIG. 7.5 – Affichage de l'état de l'unité sur l'interface

Dès qu'un joueur sélectionne plus d'une unité à la fois, des icônes s'affichent à la place des caractéristiques individuelles selon le nombre d'unités sélectionnées. Une barre de vie est également affichée pour savoir instantanément quelle unité est en danger et quelle unité est encore apte au combat. Il est possible de changer d'unité active en cliquant sur une autre icône ou avec la touche tab.

Enfin, une autre nouveauté majeure apportée fut les infobulles qui permettent d'obtenir des informations sur les ordres, compétences et bâtiments lors du passage du curseur sur l'icône correspondant, une fenêtre apparaît alors donnant tous les détails essentiels : description de l'effet, éventuel cout en mana, et touche de raccourcis entre parenthèses.

FIG. 7.6 – Affichage d'une infobulle

7.4.4 Soutenance Finale

Améliorations et derniers ajouts

Coté projet stricto sensu, j'ai procédé à quelques améliorations, par exemple j'ai introduit un temps de recharge pour les compétences afin qu'elles ne soient pas lancées de manière trop répétitive si elles sont puissantes. J'ai introduit la gestion des ressources, pour construire un bâtiment ou une unité, il est nécessaire de disposer de ressources suffisantes. Chaque joueur possède un revenu naturel de ressources à chaque seconde, mais pour former une véritable armée, il se doit d'augmenter ses sources de revenus. Les ressources s'obtiennent en construisant d'abord le bâtiment adapté pour chaque race, puis en envoyant des unités s'entrainer dans ces bâtiment, les unités en entrainement augmentent le revenu du joueur à chaque seconde, la valeur bonus apportée dépend des capacités de l'unité. Une unité en entrainement ne peut rien faire d'autre, et elle n'apparait plus sur le terrain. Une unité peut quitter ce mode à n'importe quel moment en cliquant sur son icône dans la barre de statut du bâtiment.

FIG. 7.7 – Ressources disponibles pour le joueur et population.

Un travail en symbiose avec Vladimir nous a permis de supprimer de nombreux bugs et dysfonctionnements concernant les ordres et les déplacements des unités. Un important travail s'est également déroulé en coopération avec Alban afin de mettre au point les derniers aspects indispensables de l'envoi et de la synchronisation des données, qui se devait efficace, par le réseau. Nous possédions déjà une base solide et des expérimentations avaient été effectuées, nous avons donc apportés quelques modifications et adaptations afin de gérer entièrement par le réseau ce qu'il manquait comme certains ordres, les dégâts et le lancement des compétences. J'ai également effectué un nettoyage important sur mon code, en clarifiant certaines parties et en supprimant certaines maladresses.

Pour l'interface, j'ai affiché le temps de recharge restant par dessus les icônes, une compétence en rechargement s'affiche grisée De plus si une unité ne possède pas assez de mana, la compétence s'affiche avec un filtre de couleur rouge.

J'ai apporté la touche finale à l'interface, par exemple l'affichage de la liste de production d'un bâtiment avec possibilité d'annulation. L'affichage des ressources disponibles pour le joueur ainsi que la population actuelle (nombre d'unités). Le coût en ressources pour les bâtiments et les unités via les infobulles.

FIG. 7.8 – Liste de construction d'unités.

7.4.5 Mes impressions

J'ai été heureux de réaliser ce projet très intéressant avec mon équipe. J'ai toujours été passionné par les jeux vidéos mais je m'étais toujours trouvé du coté du joueur, jamais du côté du développeur.

Ne possédant aucune expérience de la programmation au départ, j'ai du tout apprendre, je n'avais non plus aucune idée de comment un jeu était réalisé, mais grâce aux cours dispensés, à divers tutoriaux et bien sur avec l'assistance de Vladimir et Christopher, j'ai tout appris petit à petit.

J'ai également appris à travailler rapidement et efficacement afin de respecter une date limite, mais cela m'a aussi appris à travailler en équipe et surtout à tenir sur un projet de longue haleine comme celui-ci.

L'ambiance dans le groupe était bonne et axée sur le travail et la coopération lors de nos préparations de soutenances mais dans la bonne humeur et surtout avec la motivation sans cesse renouvelée de donner le meilleur de nous même afin de réaliser un jeu de qualité.

Chapitre 8

Conclusion

C'est ici que s'achève le rapport. Mais ce n'est pas la fin de l'aventure, loin de là ! Le jeu maintenant devenu jouable, il va falloir le distribuer et créer une communauté de joueurs. Pour cela nous plonger dans la communication, être à l'écoute et travailler sur des patchs de réglages de bugs mais aussi de contenu.

Nous avons également une lourde responsabilité : nous allons représenter EPITA lors du Festival du Jeu Vidéo fin Septembre. Nous espérons vous y voir, et en attendant jouez bien !

Annexes

FIG. 8.1 – Capture d'écran du jeu

FIG. 8.2 – Capture d'écran du jeu

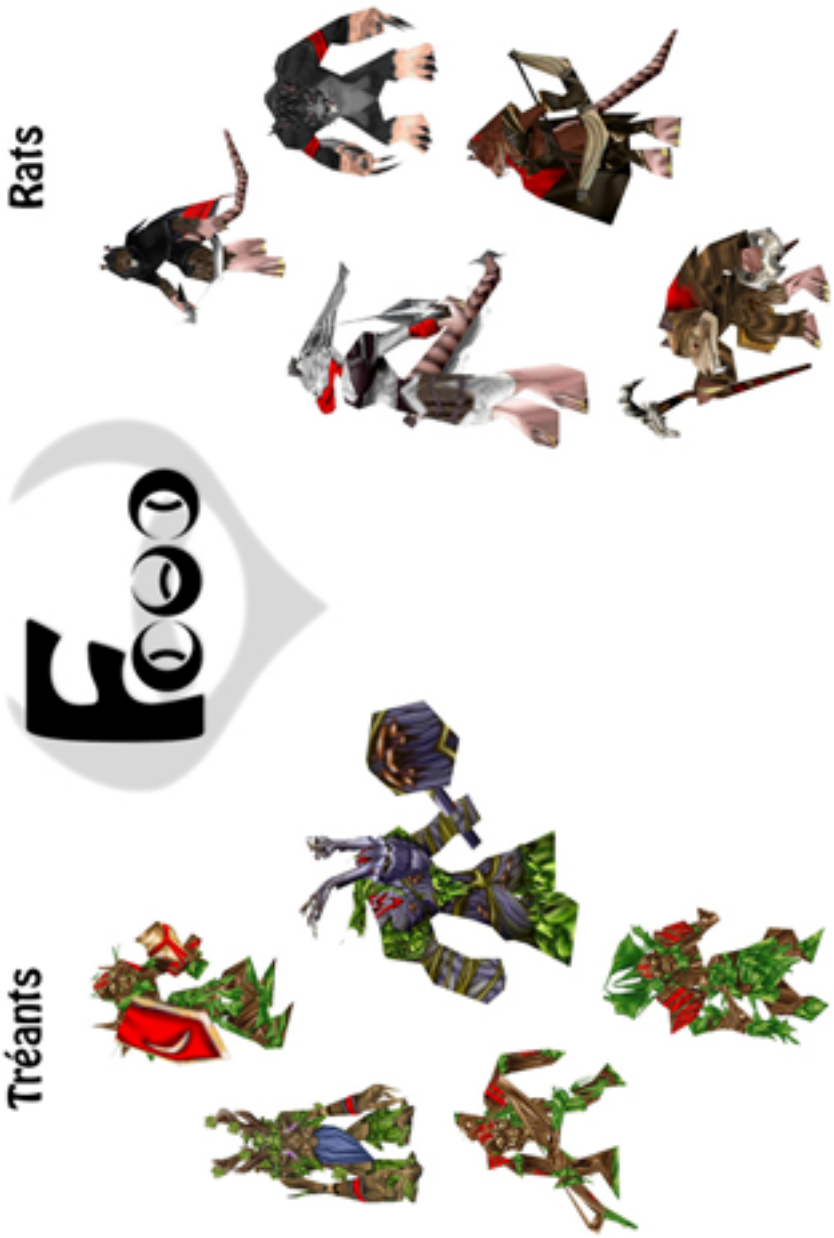


FIG. 8.3 – Unités du jeu