Project : HGF

Rapport de soutenance

troisième soutenance, le 6 mai 2008

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

Table des matières

Chapitre 1

Introduction

Les deux mois qui suivirent la deuxième soutenance furent trés prolifiques pour l'avancement du projet HGF. Nous avions réuni lors de celle ci toutes nos parties au sein d'un même programme et ainsi réalisé les bases du jeu. C'est maintenant la jouabilité qui a guidé notre développement. Qui dit jouabilité dit interaction, c'est donc grâce à l'interface que nous avons pu avancer.

De fortes relations ont été mises en place entre les membres du projet afin de pouvoir travailler. On peut citer Alban, Christopher et Félix qui ont su communiquer afin d'apprendre une structure nouvelle pour tous, mais aussi Alban, Félix et Vladmir qui ont coordonnés leurs efforts pour tout ce qui concerne les actions 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.

Chapitre 2

Marketing

2.1 Site web

Outre les différentes mises à jour liées aux actualités de l'avancement du projet, le site web[1] a subi une seule modification majeure : l'ajout des versions Espagnoles et Allemandes. Elles viennent donc s'ajouter aux Françaises et Anglaises préexistantes. L'effort technique a encore une fois été moindre, et ce grâce au système de délocalisation du texte au format XML.

Le travail de traduction n'étant pas négligeable, nous arrêterons notre extension linguistique à ces quatre versions, qui sont amplement suffisantes pour un projet de cette envergure. Enfin, nous avons acquis le nom de domaine fooo.fr, afin de continuer notre extension sur le net.

2.2 EpiPub

La régie publicitaire EpiPub a elle aussi évolué : le design du site a été épuré, et plusieurs projets se sont ajoutés à la liste, notamment un projet de la promotion 2011. Le moteur de sélection de bannière est en phase de modification afin d'inclure le nombre d'affichages de la publicité en paramètre de la fonction aléatoire, afin de ne pas pénaliser les sites de grande fréquentation.

[1] fooo-team.com

Chapitre 3

Son

Le cahier des charges stipulait que le travail sur le son serait réparti entre tous les membres du groupe, et cela est dû à notre intention de faire participer chacun des membres à l'enregistrement des différents sons.

Alban s'est cependant désigné responsable du son, en se chargeant de l'implémentation technique du son dans le jeu.

Pour ce qui est de l'intégration dans le jeu, la majeure partie du travail est terminée. Après quelques recherches, il a semblé judicieux de retenir la bibliothèque FMod[1] : 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.

[1] 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.

Pour ce qui est de l'enregistrement des voix, je suis actuellement à la recherche d'un moyen technique de la meilleure qualité qu'il soit, afin de composer une bande son acceptable pour un jeu.

Chapitre 4

Interface

Les bases de l'interface ont été posées lors de la seconde soutenance grâce aux chargements des fichiers XML et Lua qui étaient ensuite rendus dynamiquement.

4.1 Améliorations et Debuggage

Le développement des modules de l'interface, plus poussé que les deux applications déjà réalisées qu'étaient le champ de texte et le placement des boutons d'actions, ont mis en lumière des bugs au niveau du chargement et du rendu et permit d'ajouter des améliorations.

4.1.1 Héritage

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.

FIG. 4.1 – Héritage d'un élément virtuel

4.1.2 Polices

Une autre amélioration importante pour le développement était de pouvoir placer relativement des textes entre eux. La taille du texte est maintenant calculée à chaque fois que celui-ci change. Dans l'exemple suivant les bonus sont placés directement à droite des nombres en blanc. Vous pourrez également remarquer l'utilisation de différentes couleurs pour les textes.

FIG. 4.2 – 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.

FIG. 4.3 – Différentes tailles de texte

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.

4.1.3 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.

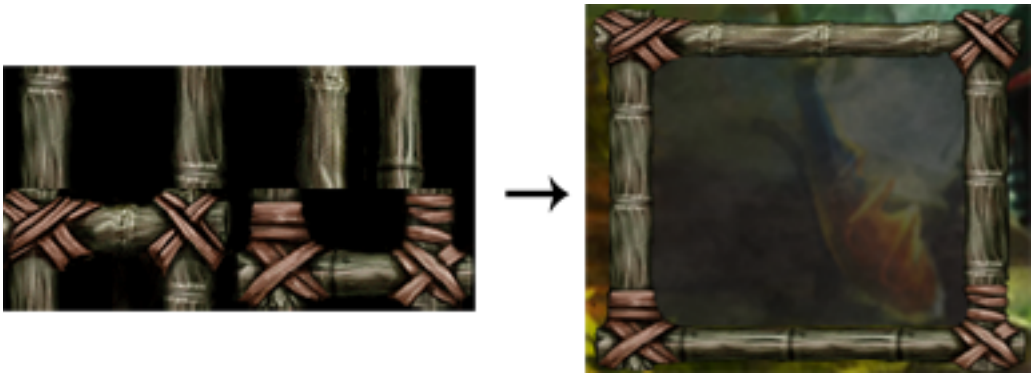

FIG. 4.4 – Affichage d'un cadre à partir d'une texture

4.2 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.

4.2.1 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.

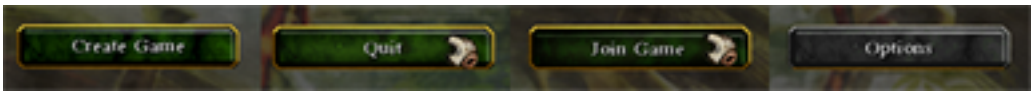

FIG. 4.5 – Différents états des boutons

4.2.2 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.

FIG. 4.6 – Liste des joueurs dans la partie

4.2.3 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.

Il n'est toutefois pas parfait. Il n'y a pas encore de gestion de limite, on peut écrire du texte à l'infini et dépasser du cadre. On ne peut pas se positionner à l'intérieur du texte grâce à un clic souris. Et une amélioration pourrait être la gestion de la sélection du texte.

FIG. 4.7 – Champ de texte pour entrer son pseudo

4.3 Conclusion

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 réussi au vu des résultats obtenus.

Chapitre 5

Réseau

Pour rappel lors de la précédente présentation du projet, il était possible via le réseau de discuter, de créer des unités et de transmettre des dommages à celles-ci.

5.1 Broadcasting

La première chose à laquelle je me suis intéressé depuis est de simplifier 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 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.

5.2 Synchronisation des données

Afin d'obtenir un jeu jouable, il était nécessaire de synchroniser les données de tous les joueurs. Ainsi, une exécution des ordres reçus par le réseau en temps réel est impossible, du fait que tous les joueurs ne recevront pas les paquets en même temps. A la réception des données, chaque ordre est donc maintenant placé dans une file d'attente (représentée par un tas), et exécuté en même temps chez tous les clients connectés au serveur.

5.3 Menu de démarrage

Je me suis enfin penché sur la réalisation du menu de démarrage : il fallait pouvoir de façon claire et simple se connecter à un serveur existant, ou en créer un. Grâce au travail de Christopher sur le XML et le Lua, il m'a été possible de scripter les pages suivantes :

5.3.1 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 (pour l'instant 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. 5.1 – Rejoindre une partie en réseau

5.3.2 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. Il permettra bientôt d'afficher les informations sur la carte sélectionnée.

5.3.3 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. 5.2 – Joueurs en attente dans un serveur

Chapitre 6

Gestion des données

6.1 Ce qui avait été realisé

Lors de la seconde soutenance, j'ai présenté de nombreuses fonctionnalités concernant l'interaction entre unités. Tout d'abord l'affichage d'un cercle de sélection coloré sous les pieds des unités sélectionnées, une fois celles-ci sélectionnées, il était possible de voir les caractéristiques tels que le nom, les points de vie, les points de mana, etc.

Les interactions entre unités étaient également possibles : poursuite, attaque, création de bâtiments avec prise en considération de la place disponible sous le curseur, utilisation de compétences avec gestion de la distance, de la cible, et des points de magie. Ces interactions sont basées sur un système générique d'ordres permettant d'en ajouter facilement de nouveaux et de créer une file d'exécution afin que l'unité en accomplisse plusieurs à la suite.

Pour faciliter la manipulation des unités, le joueur pouvait mémoriser des groupes de sélection et les rappeler aisément par un simple raccourci, améliorant ainsi la gestion de plusieurs groupes d'unités.

Tout ce travail s'est déroulé en parfaite coopération avec Vladimir et Christopher afin de réussir la convergence entre nos parties, c'est-à-dire entre la gestion des données, la recherche de chemin, le moteur graphique et l'interface.

Quant au réseau, c'est-à-dire la partie d'Alban, notre travail coopératif avait rendu possible la création d'unités par le réseau ainsi que les dégâts et la destruction d'unités.

6.2 Les nouveautés

6.2.1 Compétences

Effets périodiques et de zone

Les effets des compétences ainsi que la gestion de la cible, de la durée, de la distance étaient déjà pleinement fonctionnels. J'ai cependant rajouté de nouveaux effets. Tout d'abord une meilleure gestion de la durée du sort, avec notamment des sorts ayant un effet périodique. Il suffit de préciser dans le XML un intervalle d'effet (un *tick*) pour la compétence ainsi que les effets périodiques déclenchés (pouvant être différents des effets instantanés au lancement du sort). Il est aussi possible que l'intervalle soit nul, c'est-à-dire que l'effet périodique se rafraîchit en permanence.

Les compétences peuvent maintenant avoir un effet de zone (en précisant la distance), ainsi en combinant avec un intervalle nul, cela peut être utile pour créer des auras, par exemple un sort ralentissant toutes les unités ennemies a portée, ainsi dès qu'elles passent hors distance, l'effet s'arrête immédiatement.

Les compétences peuvent également être permanentes (durée infinie), et on peut spécifier que l'effet de la compétence est inné, c'est-à-dire qu'il sera actif des la création de l'unité sans avoir besoin de lancer la dite compétence.

Effets particuliers

De plus, afin qu'il soit possible de concevoir certains sorts particulièrement impressionnants et étant donné qu'il ne serait pas possible de rendre compte de ces effets par quelques champs dans un XML, j'ai ajouté la possibilité pour certaines compétences d'appeler une procédure Delphi en utilisant un *callback*, ainsi l'effet de la compétence appellera cette procédure en plus de ses effets. Cette procédure se chargera des effets visuels, des effets sur les unités, etc.

Il suffit pour cela de remplir le nom de la procédure dans le champ *special* du fichier XML pour l'effet instantané ou périodique.

File de construction

Les bâtiments peuvent désormais créer des unités. Pour cela des boutons avec les icônes des unités disponibles apparaissent.

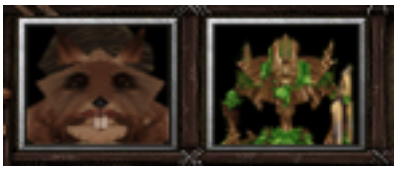

FIG. 6.1 – Choix de construction entre deux unités

Pour que le joueur puisse planifier ses actions à l'avance et concentrer ensuite son attention ailleurs sur la bataille, il est aussi possible de lancer une file de production dans un bâtiment. Les unités seront crées dans l'ordre demandé et elles apparaitront dans la position libre la plus proche du bâtiment.

FIG. 6.2 – Les unités sont crées à la suite sur les emplacements disponibles

6.2.2 Projectiles

Auparavant, les unités pouvaient s'attaquer au corps à corps mais pas à distance. Cela est désormais possible, lors d'une attaque à distance, un projectile apparaît et se dirige vers la cible à une célérité élevée. Les projectiles peuvent avoir une trajectoire droite ou une trajectoire parabolique pour les projectiles de type boulet de canon. Afin d'obtenir un résultat sensé, si l'unité est en déplacement, une interpolation sur sa position actuelle et sa vitesse de déplacement est effectuée afin que le projectile suive le mouvement de l'unité et n'atterrisse pas sur sa position de départ.

FIG. 6.3 – Trajectoire parabolique d'un boulet de canon

Grâce à cette fonctionnalité, une unité peut esquiver les projectiles lancés sur elle en déviant de trajectoire au moment opportun, en effet les projectiles n'agissent que s'ils terminent leur trajectoire à une distance suffisante de l'unité ciblée.

De manière analogue aux compétences, les projectiles peuvent avoir un effet spécial déclenché par appel d'une procédure Delphi.

6.2.3 Interface

Ordres et compétences

Grâce au travail de Christopher sur le XML et le LUA, il a été possible de créer une interface souple, aisément modifiable et intuitive. En utilisant les icônes crées par Crazyrussian et d'autres designers sur Wc3Campaigns, les boutons ont été ajoutés dans l'interface. A la sélection d'une unité, les boutons s'affichent selon les ordres possibles pour cette unité. Une unité d'infanterie par exemple aura tous les ordres de bases : déplacer, tenir la position, attaque, stop, patrouille avec ses éventuelles compétences. Une unité de construction aura tous ces ordres de base plus les icônes de réparation et de création de bâtiments. Un bâtiment n'aura que ses compétences dans la liste des icônes, etc.

Les boutons sont bien sur cliquables, un clic change temporairement la taille de l'icône pour signifier que l'action est bien prise en compte. Selon le bouton, l'action est exécutée, après choix, ou non, de la cible.

Il est aussi possible d'utiliser les raccourcis claviers à la place de cliquer sur les boutons, des raccourcis concernant les compétences sont définis dans les fichiers XML.

Lorsqu'un ordre est actif, le contour de l'icône devient vert.

FIG. 6.4 – Affichage des boutons : ordres, compétences ; et activation d'un bouton

Statut

Afin de connaître l'état actuel de l'unité sélectionnée, ses caractéristiques sont affichées dans l'interface : ses dégâts, sa défense, ses points de vie, ses points de mana. Des images miniatures des compétences actives sont aussi affichées dans la barre de statut. De plus, si une compétence modifie les valeurs d'attaque ou de défense d'une unité, le bonus ou le malus s'affiche (respectivement en vert ou en rouge) à côté de la valeur de base.

FIG. 6.5 – Affichage de l'état de l'unité sur l'interface

Barres de vie

Au cours d'une bataille il est important de savoir rapidement quel est l'état de nos troupes. Nous avons donc développé l'affichage des barres de vie au dessus des unités. Il faut passer la souris sur une unité pour afficher sa vie. Rester appuyer sur alt active l'affichage sur toutes les unités visibles à l'écran. Les coordonnées 3d de l'unité sont projetées sur l'écran afin d'obtenir des coordonnées 2d afin de l'afficher sur l'interface.

FIG. 6.6 – Affichage de l'état de l'unité sur l'interface

Sélection

Lors de la précédente soutenance, il était possible de sélectionner des groupes d'unités. Cependant le détail des unités : vie, type, nombre...ne s'affichait pas. J'ai travaillé sur cet aspect pour que le joueur puisse contrôler plus aisément d'importants groupes d'unités.

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 contrôler directement toutes les unités de même type dans le groupe de sélection, pour cela il suffit de changer le type d'unité active avec la touche tab ou en cliquant sur un autre icône. Cette fonctionnalité peut s'avérer pratique pour lancer certaines compétences qu'un seul type d'unité posséde dans la sélection.

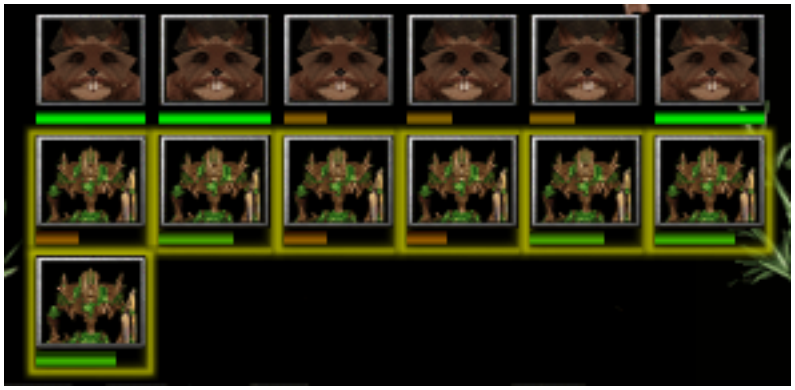

FIG. 6.7 – Multisélection et affichage des barres de vie. Les tréants sont en mode actif

Infobulles

Autre nouveauté majeure, les infobulles qui permettent d'obtenir des informations sur les ordres, compétences et bâtiments lors du passage de la souris sur un bouton. Le texte des infobulles est modifiable et lors de l'affichage du texte, les mots ne sont jamais coupés au plein milieu. Bien sur les infobulles ne s'affichent pas si une unité ne possède pas de compétences, ne peut pas construire de bâtiments, ou si il s'agit d'une unité ennemie.

Les infobulles contribuent à rendre l'interface intuitive et rapide à la prise en main. L'utilisateur peut rapidement savoir à quel ordre ou à quel sort correspond un bouton, son coût en mana et ses différents effets. Ainsi l'interface est par essence purement visuelle avec des boutons joliments realisés mais il est aussi possible de faire apparaitre des informations textuelles afin que l'utilisateur ne se sente pas perdu en voyant des icônes pas toujours très explicites pour certains effets.

FIG. 6.8 – Affichage d'une infobulle

6.3 Bilan et objectif final

Pour cette troisième soutenance, nous avons pu voir l'efficacité du travail en équipe et toutes les possibilités engendrées par la création d'un code générique pour l'interface en LUA et en XML. Ainsi de mon côté j'ai pu créer une interface intuitive, fonctionnelle et visuellement plaisante et Alban a pu créer un outil radicalement différent : un menu de démarrage du jeu.

La partie gestion des données est actuellement bien avancée, ainsi, pour la dernière soutenance, mon travail de ce côté se résumera à la suppression d'éventuels bugs, à certaines optimisations et améliorations mais surtout à la poursuite de la convergence et de l'adaptation avec le système de déplacement de Vladimir et le réseau d'Alban. L'interface se devra d'être complétée et améliorée sur certains points avec par exemple des boutons pour accéder aux options en jeu ou pour revenir au menu principal.

Chapitre 7

Moteur 3D

7.1 Introduction

Alors que la soutenance précédente a beaucoup été portée sur l'optimisation du rendu des modèles, cette fois-ci c'est leurs animations qui ont été au goût du jour.

Celles-ci ont demandé énormément de travail afin d'être finalement implantées. Dès la première soutenance du code avait été réalisé pour gérer celles-ci. Etant donné qu'il n'y a pratiquement aucune documentation sur le fonctionnement de ces animations outre la structure du modèle, il a fallu que je découvre à force de tests et de réflexion comment elles fonctionnaient. C'est maintenant chose faite.

7.2 Animations

7.2.1 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 translation, des homotéthies 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 clées ». 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.

7.2.2 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.

7.2.3 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 determinées 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. 7.1 – Animation de construction d'un bâtiment

Chapitre 8

Déplacement des unités

8.1 Recherche de chemin

8.1.1 Une meilleure intégration

Un des plus gros problèmes du pathfinding lors de la dernière soutenance était que dès qu'une recherche de chemin ne pouvait aboutir, l'algorithme parcourait toute la carte à la recherche d'un chemin. Bien que le seul moyen de vérifier qu'il n'existe aucun chemin est de tous les emprunter, il est possible de réduire les gros ralentissements causés par cette recherche négative.

Optimisations de base

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.

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

Les recherches négatives

Le deuxième problème le plus évident est 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. Il paraissait donc logique d'inclure dans la recherche de chemin un système de distance minimum à approcher, qu'il suffit de régler en fonction de la taille de l'unité, de sa destination et de la portée de son action en cours. Ainsi, dans l'exemple du réparateur de bâtiment, ce dernier ne va pas demander à aller à un point exact, mais au premier point libre correspondant à la portée de son action.

FIG. 8.1 – Le castor reçoit un chemin s'approchant suffisament de l'arbre pour qu'il puisse le construire

Cette amélioration est aussi utile pour les unités qui attaquent à distance, qui chercheront donc à s'approcher à une portée suffisante losqu'elles attaquent plutôt que de demander un chemin complet jusqu'à la cible et de s'arrêter en cours.

Le cas des unités en mouvement

Enfin, le système de recherche de chemin ne tient plus compte des unités qui se déplacent pendant ses calculs comme il le faisait avant. En effet, la seule information pour le moment disponible est la position de chaque unité au moment de la demande de chemin. Tenir compte des unités en mouvement est donc stupide, puisque cela revient à essayer d'éviter une unité qui de toutes façons a de fortes chances de ne plus être au même endroit le temps de parcourir le chemin. Pire, ce système engendre de nombreuses collisions entre les unités qui considéraient leur chemin reçu comme libre. Actuellement donc, les unités arrêtées sont toujours incluses dans les calculs, mais seulement pour éviter les collisions les plus probables et dans le cas où l'unité n'aurait pas bougé, permettre de la contourner. Mon but est de supprimer à terme toutes les positions d'unités en déplacement de la recherche de chemin globale, et plutôt de chercher à anticiper les déplacements pour plus de précision.

Les futures améliorations

Je travaille actuellement sur un système de découpage d'une carte de jeu en zones clés (cf. capture d'écran du programme page suivante). Une zone est formée d'un espace rectangulaire ne contenant pas d'obstacles naturels : arbres, falaises, etc. Ce découpage permettra de réduire les calculs de chemin en ne cherchant à chaque fois non pas un chemin d'un point à un autre d'une manière précise, mais en déterminant d'abord quelles zones traverser, pour ensuite trouver comment les traverser. Le système de recherche de chemin prendra alors 2 dimensions : le calcul à grande échelle, et le calcul précis.

Processus léger

Enfin, malgré les optimisations apportées à l'algorithme, il reste important de toujours chercher à réduire le plus possible le temps pris pour chaque calcul. J'ai donc commencé à chercher à lancer le système de recherche dans un processus léger[2] différent de celui du jeu. Ceci permettrait dans le cas par exemple de lourdes recherches de chemin, typiquement des recherches négatives inévitables, de laisser le jeu tourner de manière fluide et de faire les calculs en parallèle, sans interrompre tout l'affichage comme c'est actuellement le cas.

[2]plus connu sous le nom anglais de *Thread*

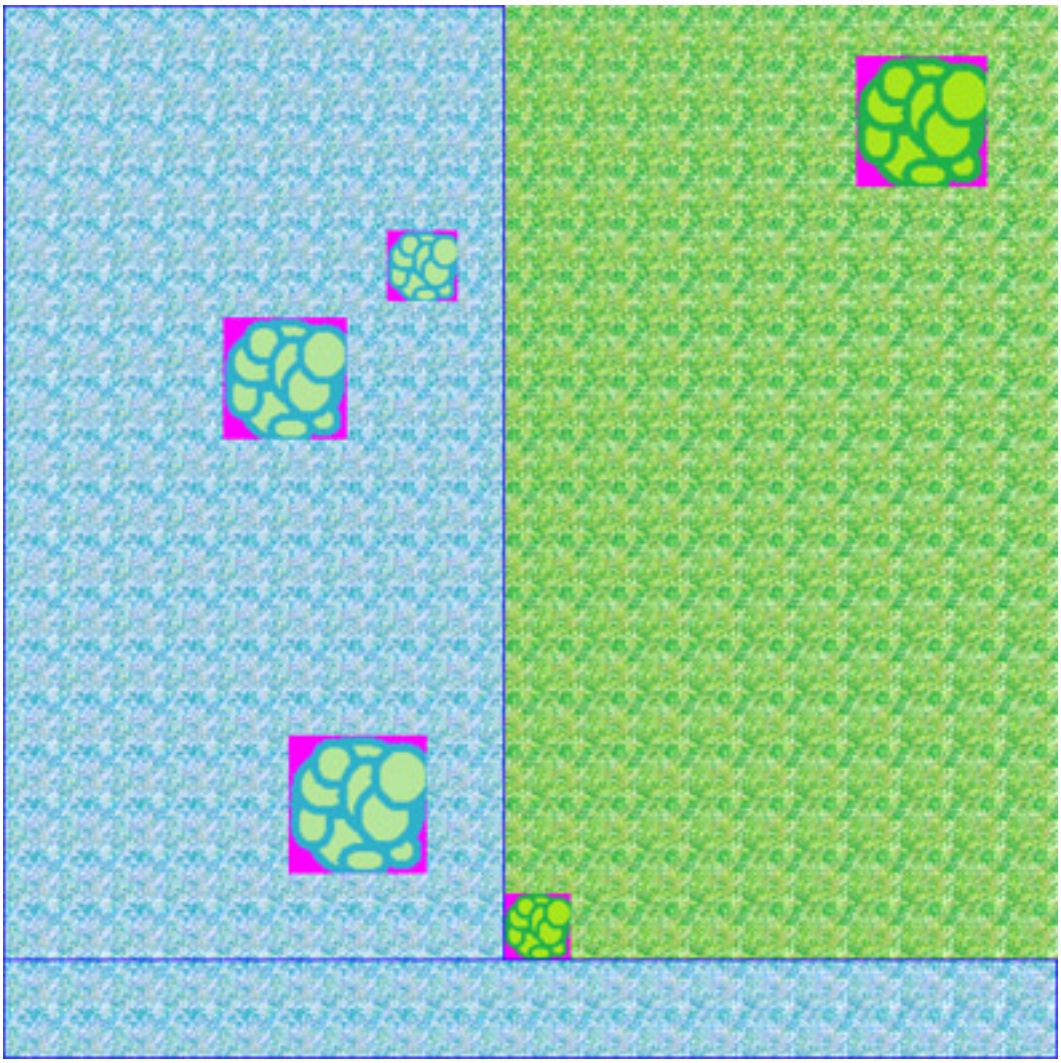

FIG. 8.2 – Ebauche de programme de découpage d'une carte selon ses obstacles

8.2 Ordres et mouvements

8.2.1 Les ordres aux unités

Agressivité

Le système d'ordres aux unités a été amélioré depuis la dernière soutenance. Il existe en effet un système d'agressivité[3] différent selon les unités et les ordres. En effet, une unité peut désormais être passive ou agressive nativement. Ainsi une unité agressive aura tendance à attaquer les unités ennemie dans son champ de vision tandis qu'une unité passive restera à sa position. 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.

Patrouilles

Chacun sait que la meilleure défense, c'est l'attaque, il existe donc un nouvel ordre agressif permettant de se défendre : la patrouille. Les unités sont en effet désormais capables de patrouiller entre plusieurs points sur la carte. Une unité en patrouille fait des aller-retours entre tous ses points de patrouillle. Il est ainsi possible de surveiller les alentours d'une zone, ou encore de balayer la carte au peigne fin avec un groupe pour trouver et détruire les derniers survivants adverses.

8.2.2 Les ordres à un groupe

Le joueur de jeu de stratégie attend de son jeu une certaine cohérence lorsqu'il est capable de gérer des groupes de plusieurs unités[4] Un ordre donné à un groupe doit alors respecter une certaine logique : Il est stupide de demander par exemple à chaque unité du groupe de trouver son propre chemin. Au contraire, lorsque l'on demande à un groupe d'attaquer une unité, on ne veut pas qu'une seule n'attaque, mais qu'elle se mettent tous contre elle[5].

[3] Oui, malgré l'appellation jeu de stratégie, il faut juste tuer les autres
[4] les groupes de un étant généralement plus simples à gérer
[5] d'où la notion de stratégie

8.2.3 Les points à améliorer

Les groupes

La notion de groupe n'est pas encore réellement présente ; pour le moment, un ordre donné à un groupe sera simplement donné à chacun de ses membres. Les groupes permettront de considérer un ensemble d'unité comme une seule entité. Un déplacement d'un groupe sera donc un déplacement d'un ensemble d'unités selon une formation, plus ou moins respectée pendant le déplacement selon les obstacles. La formation sera probablement celle de départ dans un premier temps, et éventuellement après des formations plus géométriques. Notre jeu visant à simuler des batailles entre plusieurs petites tribues ou village plutôt qu'entre des armées de milliers d'unités, Il est probable qu'un des aspect du jeu soit aussi de laisser le joueur contrôler de manière précise chaque unité, et donc de ne pas lui imposer de formations mais plutôt de lui laisser les gérer lui-même.

Autres vecteurs de recherche

Concernant à la fois la recherche de chemin et le déplacement en lui même, une des amélioration qui rendrait tout simplement le déplacement réaliste serait un système de prédiction de chemin. Il faudrait en effet lorsqu'on calcule un chemin tenir compte des autres unités autour qui risquent de rentrer en collision avec celle qui cherche son chemin. Il existe de nombreuses manières d'éviter les problèmes de collisions, la plus simple étant probablement de donner une priorité de déplacement à certaines unités pour qu'en cas de collision à venir, la plus prioritaire passe et l'autre attende.

Certains spécialistes dans le domaine du déplacement d'unités de jeu de stratégie affirment qu'il est impossible d'éviter les collisions, plutôt que d'essayer de vérifier que nous sommes plus forts qu'eux, je vais plutôt me pencher à la résolution du problème de collision une fois arrivé. Notamment avec des système de rétablissement d'une certaines distance entre les unités. Sur le principe, cet algorithme récursif ordonne aux unités, de l'intérieur vers l'extérieur, de se séparer l'une après l'autre du groupe collé.

Chapitre 9

Conclusion

Afin d'aller au plus loin dans le projet nous avons décidé de nous réunir pendant une semaine afin de travailler efficacement. Nous avons réussi à adopter une attitude professionnelle en nous fixant de nouveaux objectifs en permanence. Nous sommes donc devenus en quelque sorte une petite entreprise en train de créer un jeu vidéo.

Une entreprise a des responsabilités, la principale étant de fournir le fruit de son travail à temps et pleinement fonctionnel, ce à quoi nous allons nous atteler dans les semaines qui nous séparent de notre soutenance finale.