Ececraft

Warcraft 1

Warcraft 2

Warcraft 3

Starcraft 2

→ Principe du jeu

Le jeu qui vous est demandé est un jeu de « stratégie en temps réel » ou « Real Time Strategy » (RTS). Si vous ne connaissez pas ce genre vidéo-ludique nous vous invitons à voir (avec modération) quelques extraits de parties sur les canaux usuels de diffusions (Youtube & co) qui illustreront mieux que des images fixes ce genre d'univers et les mécanismes d'interaction. Si vous ne savez pas par où commencer jetez un œil sur l'introduction d'un passionné sur un des monuments du genre : **Warcraft 2**. Dans un registre similaire on trouve du coté du jeu sur mobile le populaire Clash of Clans.

Le type de jeu attendu n'a rien à voir avec « World of Warcraft », ce n'est pas une aventure à la 1ère personne : le joueur doit gérer des **troupes** sur un **territoire** qu'il **colonise** et **défend**...

On ne va bien sûr pas vous demander ni de développer au niveau ni de produire la masse de travail, tant en terme de contenu graphique et sonore qu'en programmation, que demande un jeu Blizzard.

ECEcraft, votre projet, sera donc une version « allégée » de ce genre de jeu. On imagine que personne avant vous n'avait eu l'idée de ce genre de jeu. Vous êtes une équipe de 3 (ou 2) ingénieurs innovants et vous devez, en 1 mois ouvrable, tout en continuant votre job (n'oubliez pas les autres matières !) produire une **maquette** qui saurait convaincre un éditeur de jeu ou des investisseurs de la viabilité du concept.

Afin de vous laissez **toute liberté** quant à l'univers, à son histoire, au style, aux réglages de « game-play », la description des aspects attendus est volontairement abstraite. Par exemple quand on dit « ennemi » il ne faut pas forcément comprendre Orcs/Gobelins etc. Les ennemis peuvent être des lapins blancs, ou des *men in black*, ou des microbes... respectivement dans un univers où vous jouez des légumes qui essaient de s'installer dans un champs en style cartoon, ou un univers où vous jouez des envahisseurs extraterrestres genre space-invader, ou un univers où vous êtes les globules blancs défendant les amygdales et les sinus d'un bébé. Votre jeu peut être réglé plus vers l'attaque (les envahisseurs) ou plus vers la défense (les globules blancs) ou plus vers la simulation d'écosystème (les légumes... ont-ils assez d'eau, de soleil, de minéraux ?). Je me permets de bien insister : les mécanismes sont imposés (pour garantir un certain niveau d'investissement en programmation) mais la nature guerrière des exemples historiques (Warcraft/Starcraft etc...) n'est pas requise, il n'y a pas forcément du sang, du jus de carotte fait aussi bien l'affaire. Il faut quand même qu'il y ait un challenge, une difficulté à surmonter : il faut jouer bien pour « gagner », sinon on perd (d'une façon ou d'une autre).

Il ne vous est pas demandé de fournir une copie conforme du jeu Warcraft ni même d'essayer.

→ Cahier des charges de base : 15 points

Le jeu se déroule dans un univers en 2 dimensions vu de dessus (facultativement en fausse perspective ou en perspective isométrique). A l'échelle des déplacements des personnages, le terrain est plus grand que l'écran. On dispose soit
- d'une zone **radar** ou vue satellite qui montre l'ensemble du terrain en tout petit, chaque personnage représenté par un seul pixel, et un cadre principal de jeu représentant une partie du terrain, on peut ***scroller*** (barre de défilement ou autre...) ce cadre pour jouer sur une autre zone du terrain
- d'un seul cadre de jeu avec possibilité à la fois de ***scroller*** (barre de défilement ou autre...) et de **dézoomer** (prendre de la distance, de l'altitude) pour voir la totalité du terrain, et **rezoomer** pour se rapprocher de l'action dans une zone particulière du terrain.

Sur ce terrain évoluent des « personnages » (des globules blancs, des men in black, des carottes...). Les personnages du joueur (ses troupes) doivent atteindre un certain objectif, par exemple s'étendre et conquérir, ou juste survivre un certain temps. Le joueur peut **sélectionner un ou plusieurs personnages**, par exemple avec un cadre souris (zone de sélection) pour leur donner un ordre, aller à un endroit, exploiter une ressource, construire, attaquer, défendre. Il n'est pas demandé de prévoir tous ces ordres : on peut doter les personnages du joueur d'une autonomie suffisante pour qu'ils prennent des initiatives, il suffit alors d'une seule catégorie d'ordre : aller vers un endroit.

Dans notre cahier des charges simplifié il n'est demandé que **2 classes** de personnages du joueur. Par exemple dans un univers médiéval « traditionnel » on pourrait imaginer une classe d'artisans et une classe de guerriers. Chaque classe dispose de caractéristiques spécifiques : chaque guerrier aura à sa création le même nombre de points de vie, le même nombre de points d'attaques, les mêmes aptitudes que tous les autres guerriers (par exemple tuer des Orcs). Chaque artisan aura à sa création le même nombre de points de vie, les mêmes aptitudes que tous les autres artisans (par exemple chercher des ressources et construire des bâtiments avec ces ressources).

Les personnages du joueur ont une mémoire (élémentaire) : ils se souviennent du dernier ordre qu'on leur à donné. **Le joueur n'a pas a prendre le contrôle d'un personnage et de le diriger dans les détails (haut/bas/gauche/droite)**, un personnage qui a reçu l'ordre d'aller à un endroit y va. Si il se fait attaquer en cours de route il peut éventuellement décider d'engager le combat et oublier sa « mission » de départ. Tout dépend des réglages de votre jeu : des personnages plutôt obéissants mais qui manquent d'initiative, qui se laissent taper dessus tant qu'on ne leur donne pas l'ordre de se défendre, ou des personnages très autonomes mais difficiles à commander ou qu'il faut souvent rappeler à l'ordre. A votre niveau on ne vous demande pas d'algorithme de recherche de chemin : si un personnage a reçu l'ordre d'aller à un endroit il essaye d'y aller **en ligne droite**, si il rencontre un obstacle infranchissable (voir plus loin types de terrain) il n'est pas demandé de le rendre assez intelligent pour le contourner. Il peut décider de s'arrêter, ou de reprendre une direction au hasard, ou de patrouiller en restant dans la zone, de même pour un personnage arrivé à destination : à vous de trouver le réglage le plus intéressant.

De nouveaux personnages peuvent **naître** et venir compléter les troupes du joueur. Les nouveaux personnages ne peuvent apparaître que dans une **construction** (ou bâtiment). Chaque naissance d'un certain type requiert une certaine quantité de **ressources** (par exemple un guerrier demandera plus de ressources en métal qu'un artisan...) et mobilise un bâtiment pendant un certain temps. La construction d'un bâtiment requiert une certaine quantité de ressources et mobilise un ou des constructeurs pendant un certain temps. Les ressources sont donc **dépensées** quand on créé des bâtiments et quand on créé de nouveaux personnages dans les bâtiments. Pour **re-gagner des ressources** il faut aller les chercher sur le terrain à des emplacements spécifiques, par exemple la ressource en bois se trouve dans les forêts, la ressource en métal dans les mines...

Pour simplifier, la gestion des ressources sera globale *(on n'a pas dit avec des variables globales)* : il n'y a pas à gérer des transports de marchandises d'un endroit à un autre, le joueur dispose d'une certaine quantité de bois, cette quantité augmente quelque soit l'endroit ou un de ses personnages coupe du bois, et il peut utiliser cette quantité de bois aussitôt pour construire un bâtiment ou faire naître des personnages où bon lui semble. Les quantités de ressources disponibles à un moment données seront indiquées au joueur d'une façon ou d'une autre à l'écran (valeurs numériques, jauges...). Dans la version minimale il est demandé de prévoir **2 types de constructions** (par exemple des casernes pour faire naître des guerriers et des écoles pour faire naître des artisans) et **2 types de ressources naturelles** (par exemple du bois et du métal).

Des personnages (troupes ennemies) viennent introduire de l'adversité dans le jeu. **Les personnages ennemis sont totalement autonomes.** Contrairement à la plupart des jeux de stratégie temps réel où l'ennemi est doté de la même structure hiérarchique et des même contraintes que nous (jeu à égalité), et est joué par un ou plusieurs autre joueur humain adversaire (ou une IA assez évoluée),

notre projet se limitera, dans sa version de base, à **une seule classe ennemie**. Les ennemis n'ont pas à se structurer en villages, n'ont pas besoin de construire des bâtiments pour naître, ne disposent pas d'une intelligence artificielle avancée. Dans la version de base ont peut les considérer comme de simples monstres errants, qui apparaissent au hasard dans le monde (ou dans des zones spécifiques, par exemple des marais) marchent au hasard et attaquent tout ce qui bouge (vos troupes!). Leur fréquence d'apparition doit être soigneusement réglée de telle sorte que le joueur est obligé de se défendre (ou d'attaquer), mais a ses chances (il n'est pas submergé). **Les ennemis peuvent tuer** ou faire disparaître les persos du joueur, quelle que soit la forme plus ou moins violente de la disparition.

Les « combats » peuvent être gérés au contact (collision) ou à distance (projectiles) ou encore ne pas ressembler du tout à des combats. On peut imaginer par exemple un jeu où le but est uniquement de construire des fortifications assez rapidement pour se mettre à l'abri d'une vague de loups garous. La fréquence d'apparition des loups garous augmente en cours de partie, la partie est minutée, et pour survivre 5 minutes il faut fortifier le village aussi vite que possible, les ennemis sont immortels, il faut juste les bloquer.

Enfin votre terrain doit proposer des zones spécifique avec au moins **4 types de terrains** par exemple des zones inaccessibles (ronces, montagnes, eau) aussi bien aux troupes du joueur qu'aux ennemis, des zone neutres accessibles et constructibles (plaines) des zones avec la 1ère ressource (forêt) et des zones avec la 2ème ressource (mines). Ce n'est qu'un exemple, on peut imaginer que les ennemis ne sont pas bloqués par la forêt et les troupes du joueur savent nager...

Au démarrage d'une partie, le joueur dispose de personnages initiaux, en quantité et à des endroits bien précis (toujours les mêmes) du terrain, éventuellement autour de quelques constructions initiales. Le but du jeu est alors, en général, de se développer, par exemple partir avec 3 artisans perdus au milieu de la forêt et terminer avec 10 casernes et une armée de 100 guerriers. Mais comme nous encourageons l'originalité il est possible de renverser cette logique : partir avec 100 citadins dans un centre commercial et arriver à en sauver au moins 3 après 5 minutes d'une attaque de zombis... Dans tout les cas le CDC de base impose de voir **à un moment donné au moins 100 personnages gérés simultanément**, répartis selon les diverses classes (classe 1, classe 2, ennemis). Et bien sûr pour des raisons évidente de difficulté (donc d'intérêt) de programmation, il est **indispensable que ce nombre soit variable (des personnages apparaissent, des personnages disparaissent)**

Enfin dans ce CDC de base il y a la possibilité de disposer d'un menu d'accueil, de pouvoir choisir **un terrain parmi 3**, de **pouvoir retourner au menu à tout moment soit sur un game-over soit sur un abandon soit sur une victoire**, et de pouvoir **relancer une partie** soit sur le même terrain soit sur un des 2 autres terrains proposé. En résumé on veut pouvoir rejouer, indéfiniment, sur les 3 terrains de jeu, **sans avoir à quitter/relancer** le programme, et **sans appels récursifs au *main*** (demandez...) Compte tenu des impératifs de la présentation lors de la soutenance un terrain doit pouvoir être joué en à peu près 6 minutes, ce qui est très court pour ce genre de jeu. Il ne sera sans doute pas possible de voir en intégralité les 3 terrains, mais les 3 terrains doivent être jouables. Dans le CDC de base il n'est **pas** demandé de proposer 3 thèmes graphiques distincts : les 3 terrains peuvent avoir les même dessins (d'arbres, de plaine, de personnages...) c'est juste la cartographie qui change.

Le CDC de base étant assez ouvert, n'hésitez pas à nous faire part de vos idées : si c'est intéressant à programmer nous sommes partant. Si c'est pour diminuer l'exigence de programmation, non...

→ Résumé du CDC de base :

2 types de personnages dans les armées du joueur (apparence, coût et caractéristiques spécifiques)
2 types de constructions dans les villages du joueur (apparence, coût et caractéristiques spécifiques)
2 types de ressources naturelles (apparence/lieux spécifiques du territoire) Gestion « globale »
4 types de terrains (reconnaissables et avec des caractéristiques spécifiques)
1 type d'ennemi (qui arrive par les frontière du territoire ou naît aléatoirement ici ou là)

→ Extensions : 4 points

Le jury évaluera le CDC de base sur tous les points sus-mentionnés, tout manquement ou réalisation trop approximative sera pénalisé au barème de 15 points prévu. Par exemple si par commodité pour gérer différents types de terrain votre territoire de jeu est découpé en cases grossières et que les personnages se déplacent de case en case comme des pions au jeu d'échec au lieu de translater en douceur en s'orientant du bon coté, l'aspect déplacement sera considéré comme imparfaitement maîtrisé. Le barème des extensions sur 4 points est séparé : vous ne pouvez pas compenser des manquements au CDC de base en collectionnant les extensions au delà de 4 points. Il y a une note sur 15 d'une part et une note sur 4 d'autre part. Le jury se réserve le droit de ne pas considérer du tout les extensions si le CDC de base est trop déficient.

La liste suivante n'est pas exhaustive, il n'est pas nécessaire de réaliser toutes ces idées pour obtenir les 4 points d'extension, discutez avec votre chargé de TP des autres idées que vous pourriez avoir ! Les points (ou ½ points) accordés par le jury dépendront de la difficulté et de la qualité de réalisation.

Animations : les personnages sont animés (séquences d'images, au moins rudimentaires)

Animations spécifiques : les personnages ont des aspects/animations différents selon leur activité

Carte à zones inconnues : seules les parties du territoire visitées (par un personnage du joueur) apparaissent sur la carte, les autres zones restent blanches ou noires : terra incognita !

Carte radar à visibilité limité : seuls les ennemis dans le champ de vision d'un personnage du joueur apparaissent sur le radar, les autres n'apparaissent pas (ce qui ne les empêche pas d'exister!)

Attaques à distance, projectiles : les personnages du joueur et/ou les ennemis se battent à distance, on voit voler des flèches ou des boulets ou des rayons lasers... Bonus pour la modélisation de trajectoires balistiques, comme la parabole d'un projectile de catapulte vu en pseudo-3D...

Effets visuels évolués, effets de particules : dans Warcraft et plus encore dans Starcraft un soin particulier est donné à la représentation des zones d'attaque, flammes, auras, explosions…

Trajectoires avancées : Les personnages et/ou ennemis savent contourner les obstacles par le plus court chemin (ou presque)

Autonomie avancée : L'intelligence artificielle des ennemis et/ou personnage du joueur mérite son nom, il y a des réglages fins entre le hasard et le déterminisme rendant les comportement plus intéressant, moins prévisibles, plus riches, tout le monde prend des initiatives, il y a des comportement collectifs, des heuristiques...

Ressources avancées : l'exploitation des ressources conduit à une altération irréversible du terrain, par exemple là où on a coupé du bois on voit que les arbres sont coupés (et on ne peut plus en couper là)

Types supplémentaires : vous pouvez ajouter d'autres types de personnages ou d'ennemis, de bâtiments, de ressources, de terrain. Discutez avec votre chargé de TP de la valorisation en fonction de ces ajouts (ce n'est pas forcément un point de gagné par classe de guerrier en plus !)

Ennemi structuré : l'ennemi n'est pas juste une bande de monstres errants sortant des marais, c'est une armée comme vous ! Avec plusieurs classes de personnages, des constructions etc... (Warcraft...)

Pseudo 3D ou perspective isométrique : plutôt que d'avoir une vue strictement plongeante (verticale) on peut essayer dans les graphismes d'avoir un effet de vision inclinée. Sans aller jusqu'à la "vraie 3D" avec lignes de fuite (rétrécissement des objets éloignés) on peut assez facilement faire de la perspective de style isométrique ou suggérer une vue non verticale. Veuillez consulter l'exemple 4 du cours 0 (dans le .zip du cours 0) pour avoir un exemple de construction isométrique, et l'exemple 4_3 du cours 0 pour avoir un exemple de vue « plongeante non verticale » à la Zelda génération GBA

Clash of Clans (Supercell): un exemple de perspective isométrique

Effets sonores interactifs : des bims, des boums, des arghh ! Bien synchronisés avec l'action bien sûr. Il est aussi possible d'ajouter une musique de fond (voir format .ogg avec Allegro 4.4.2) mais n'espérez pas avoir beaucoup de points pour juste un seul load_sample et play_sample...

→ Originalité et cohérence : 1 point (+1 point bonus exceptionnel)

Le jury appréciera l'originalité de votre proposition. Essayez de déplacer la mécanique de jeu imposée dans un univers différent, adaptez le récit et le *gameplay* à cet univers pour en faire un tout cohérent et harmonieux. Dans le point d'originalité est aussi compté (sans caractère obligatoire) un travail de design graphique personnel, valorisé par rapport à la simple utilisation de graphismes empruntés (voir ci-dessous).
Très exceptionnellement 1 point supplémentaire de bonus est laissé à l'appréciation du jury (talent prodigieux à la palette, images de synthèse sensationnelles...) Attention ceci ne doit pas vous faire oublier qu'il s'agit avant tout d'un projet de programmation : chaque membre de l'équipe doit participer au code, pouvoir expliquer en détail les parties sous sa responsabilité et de façon synthétique le reste du code source, sans se retrouver perdu.

→ Consignes générales :

- Il est autorisé d'utiliser des graphismes tiers trouvés sur Internet, si possible libres de droits (*Creative commons*...) ou avec *Copyright,* dans ce dernier cas votre logiciel doit rester strictement dans le cadre scolaire (citation dans un travail d'étude) et ne doit pas être publié. Dans tous les cas les sources tierces doivent être rigoureusement citées dans un fichier **licence.txt** dans votre archive.

- Il est autorisé d'utiliser/adapter les codes sources publiés sur le site principal du cours, ainsi que des **extraits** de code source libre de droit (*public domain*, GPL, LGPL …) en précisant ces emprunts. Les commentaires initiaux de ces codes sources empruntés doivent être mis à jour par vos soins pour refléter leur utilisation dans votre contexte (appropriation correcte des codes tiers utilisés). En soutenance vous devez être capable d'expliquer chaque ligne de code de votre projet déposé.

- Il n'est pas autorisé d'utiliser une librairie complète ou un « *framework* » en dehors des versions proposées de Allegro sur le site principal du cours (4.2.2 ou 4.4.2) et des bibliothèques standard du C (stdio, stdlib etc...). Il n'est pas autorisé de copier massivement un code tiers : l'objectif est de réaliser votre moteur de jeu, pas de se brancher sur un moteur de jeu pré-existant. **En dehors des exemples du cours, toute détection de copie massive de code tiers sera considéré comme plagiat et très sévèrement sanctionné, avec circonstances aggravantes si**
  - l'emprunt n'est pas cité
  - l'emprunt vient du travail d'une autre équipe d'étudiants de l'ECE pour ce même projet
  - il y a tentative de dissimulation (maquillage de code...)

- Le code source doit être présentable : bien indenté, raisonnablement aéré et commenté. Pour rappel, un bon commentaire ne paraphrase pas le code (// xp prend la valeur 10) mais explique à quoi sert le code (// L'abscisse du perso est mise à gauche de l'écran). L'ensemble doit être structuré : sous-programmes regroupés dans des .c et .h par thématique, structures de données adaptées aux problèmes à traiter, constantes nommées, éviter autant que possible les duplications de codes similaires au profit de boucles, tableaux, sous-programmes, paramètres.

→ Contrainte de volume

Le dépôt à faire sur campus ne devra pas dépasser **50Mo zippé**. Allegro 4.2.2 ne prend pas en charge les formats images compressés, les images .bmp peuvent rapidement peser très lourd. Surveillez ce paramètre. Ce sont surtout les images de fonds et « cartes de collision » qui pèseront le plus si vous en utilisez. Avec la version 4.4.2 mise à disposition sur le site principal du cours vous pouvez utiliser des images en format .png (préférable pour les graphismes « lisses » et pour conserver des couleurs exactes : cartes de collisions) et/ou des images en format .jpg (à privilégier pour des images bruitées comme des photos numériques). Il ne s'agit pas d'une obligation, vous pouvez rester sur du .bmp si le volume ne pose pas de problème.

Voir les options de compilation (Project → Build options... → Linker settings) de l'exemple de projet check_alleg_4_4_png_jpg de allegro_checking.zip pour compiler un projet capable d'utiliser les formats .jpg et .png.

→ Travail à faire

Le projet est à faire en trinôme ou binôme, à l'intérieur d'un même groupe de TP. Les consignes de constitution des équipes de projet seront communiquées en temps utile via le forum.

Vous réaliserez ce projet en respectant la méthode de conception Données/Traitements/Interface apprise : conception puis réalisation. Tenez à jour, **dès le départ de la conception**, un « cahier de laboratoire » : vous remettrez, à l'occasion de la soutenance, un **bilan écrit** par équipe qui contiendra des schémas de vos structures de données (structs, tableaux, listes chaînées éventuelles...), des schéma géométriques (repères, vecteurs) seront appréciés, votre ACD, le graphe d'appels de vos sous-programmes en précisant les entrées et les sorties, quelques captures écrans, ainsi que le bilan collectif du fonctionnement de votre équipe, et le bilan individuel de votre implication, de vos difficultés et de vos progrès via ce projet.

→ La date de soutenance est fixée pour la semaine du 1er Mai, la date exacte et les modalités de rendu vous seront précisées rapidement. Vous aurez une 1ère séance de TP orienté projet dès la semaine du 27/03, durant laquelle vous pourrez poser toutes vos questions à votre chargé de TP et commencer à pratiquer les techniques de programmation spécifiques au projet.

Robin Fercoq et toute l'équipe encadrante,

20/03/2017

Version 1, sujette à ajustements mineurs en cas de besoin.