DEVELOPPEMENT :

1) Structures:

Nous avons mis en place plusieurs structures de données tel que des structures et des énumérations.

En effet, chaque case de notre terrain de jeu est composé d'une structure case qui est composé des coordonnées de l'image qu'elle comporte pour pouvoir traiter clics souris et pouvoir afficher les images au bons endroits. Cela permet également de très facilement déplacer toute la grille de jeu grâce a la caméra en changeant ces variables propre a chaque case. La structure case contient aussi le type de case qui définira le l'image de la case et ses propriété rattachées. On stocke aussi un pointeur sur structure pièce et un pointeur sur structure bloc qui valent NULL de base sur une case (pas de pièce ni de bloc sur la case). Enfin on sauvegarde l'état de la case quant au possible déplacement, attaque et a la pose de bloc sur cette case qui mettent une des trois variable à un ou zéro selon si la case est apte a recevoir une de ces actions(cela permet aussi l'affichage de cases possible pour l'utilisateur).

La structure pièce contient les statistiques d'une pièce avec une certaine classe défini par une énumération. Ces statistiques évoluent tout au long de la partie et chaque pièce est défini comme expliquer plus haut sur une case de la grille. Une variable a aussi été mise en place pour vérifié sur la pièce est sélectionné ou non ce qui permet d'effectuer un panel d'action avec la pièce tel que se déplacer ou attaquer.

La structure bloc quant a elle comporte les statistiques du bloc, l'id du joueur possédant ce bloc et une variable utile pour la sélection comme avec les pièces vu ci dessus.

Pour finir avec les structures principales, nous en avons une appelé joueur\_t servant a stocker les id de chaque joueurs avec leur nombre d'unités, le fait qu'un joueur est un bot ou un humain et leur points d'action. Cela permet par la même occasion d'éliminer facilement les joueur ayant perdu en regardant leur nombre d'unité et en mettant leur id à -1 si ce nombre est réduit a zéro.

2) I.A.

L'I.A est une grande partie de notre projet, elle a permis par exemple de faire des simulation de plus de 5000 joueurs non humain les uns contre les autres a grande vitesse pour tester les fonctions de pathfinding, de combat, déplacement etc. a grande échelle pour déceler chaque problème de conception.

Notre I.A est basé sur les réflexions effectué sur la stratégie la plus intéressante a adopter selon nous du faite que le nombre de possibilité a chaque tour est trop élevé pour permettre de réaliser un algorithme permettant de voir plusieurs coups en avance.

Elle consiste tout d'abord en la sélection d'une unité aléatoire parmi ses unités puis l'unité sélectionné regarde le nombre d'ennemi et d'allié a portée pour choisir si elle effectuera un déplacement vers un allié, vers un ennemi ou bien une attaque.

dans le cas ou cet unité est un prêtre est un prêtre, si un allié a porté possède moins de 25 points de vie elle le soigne sinon l'unité parcourra la grille en partant de chaque allié a portée puis en étendant cette recherche de la portée de l'unité ennemie avec la portée maximal pour compter le nombre d'ennemi potentiel qui pourrais attaquer cet allié au prochain tour. L'unité soigne alors cet allié.

Dans le cas ou l'unité sélectionné n'est pas un soigneur, elle attaquera l'ennemi a porté avec le moins de points de vie.

Concernant les déplacements, L'I.A. possède une fonction de pathfinding a longue distance permettant de se rapprocher a chaque tour d'une cible défini(allié ou non) même si celle ci est de l'autre coté de la carte et non atteignable en un déplacement direct. C'est un déplacement sur plusieurs tour qui peut évolué entre deux tour si un ennemi se rapproche trop par exemple.

A la fin de chaque tour (quand il ne reste plus de points d'action au bot qui joue) l'I.A. sélectionne chacune de ses unités pour orienter leur direction de block dans le coté de la carte comportant le plus d'ennemi.

3)Réseaux

Willhem

4)Boucle principale

Au début de la partie les Menu permettent de sélectionner différentes variables puis quand cela est effectué et que les unitées voulues pour chaque joueurs sont choisie la partie débute.

La partie commence par une initialisation complètent de la grille de jeu, des unités, des joueurs, une création du fichiers de logs en rapport avec la partie inscrivant l'heure et la date du début de la partie et préparant toutes les variables utile lors de la partie.

Apres cet initialisation la partie se décompose en trois étapes principales qui sont l'attente d'action utilisateur, l'action a effectuer et l'affichage a la fin de chaque boucle.

Si le joueur actuel est un bot l'I.A. le fait jouer avec une certaine vitesse qui a été défini au début de la partie. Pendant ce temps il est impossible aux joueurs humain d'effectuer la moindre action excepté la mise en pause qui permet par exemple de quitter la partie et de retourner au menu.

Cependant si le joueur actuel est un humain, la boucle attendra des actions souris ou clavier avant d'effectuer la moindre action. Il faudra par exemple, pour un déplacement, sélectionner une unité puis cliquer sur une des cases en surbrillance bleu pour s'y déplacer.

Chaque action coute un points d'action au joueur actuel et il peut faire des actions jusqu'a ce que le montant de point d'action soit égale a zéro. Une fois ce montant atteint, si le joueur actuel est un bot, ce sera au tour du joueur suivant sinon s'il est humain il faudra attendre l'action de fin du tour.

Seule l'affichage a lieu a chaque itération de la boucle. Il permet de mettre à jour toute l'interface graphique ainsi que la grille en fonction des actions qui ont été effectué ou en fonction de la taille de la fenêtre de jeu qui a pu être modifié. cette une seule fonction d'affichage qui gère tout cela mais elle appelle des fonction pour l'affichage des animations, des texte animé et des images et texte de base.

A la fin de ces trois principales étapes des variable servant aux animations, au système jour nuit etc. sont mis a jours. Si la fin d'un tour a été détecté le jeu passe la main au joueur suivant en passant les joueurs éliminés.

5)Combat

Willhem

6)Affichage via SDL

Notre fenêtre de jeu est initialisé avec la taille maximale possible pour l'écran utilisé puis il est possible de la redimensionner a souhait. En effet nous utilisons des pointeur sur int qui sont utilisé dans la grande partie des images et texte pour que l'affichage soit le plus responsive possible ainsi l'affichage évolue en même temps que le changement de la taille de la fenêtre.

Apres l'initialisation de la fenêtre et des différente bibliothèque de SDL nous chargeons via une fonction nommé load\_image toute les textures des différentes images qui seront utilisé dans un tableau de SDL\_Texture. Cela permet de ne pas avoir a recharger les textures à chacune de leur utilisation. Toute les images ont été faite par un logiciel nommé graphic gales.

Nous avons deux fonctions principale d'affichage, une première qui fait un affichage "vu du dessus" et une seconde avec une vu en 3D isométrique. La première étant une version basique de la seconde que nous utilisions avant et qu'il est possible d'afficher en changeant une option dans le menu au début de la partie.

L'affichage présente plusieurs plan en fonction du moment ou l'image ou le texte est rentré dans le renderer qui servira a afficher l'image. Au premier plan il y a l'interface, au second les buissons ou rocher, au troisième plan les personnages, au quatrième plan les unités, au cinquième plan les arbres, montagnes et bâtiments, au sixième plan la surbrillance de différente couleur pour le déplacement, l'attaque ou le placement des blocs et enfin au dernier plan les cases elles même.

La fonction d'affichage fonctionne grâce a une fonction pour le texte, une fonction pour les animations de texte et une fonction d'affichage des images et de leurs animations.

Les animations sont simplement une grande images que nous séparons en plusieurs partie grâce a des SDL\_RECT que nous rentrons en paramètre de la fonction d'affichage d'images.

Les images tel que les arbres, les montagnes et les bâtiments possèdent une version transparente pour permettre a voir les unité a l'intérieur des bâtiments ou derrière les arbres et montagnes. Ces images devienne transparente aussi quand la case ou elles sont affiché est apte à une quelconque attaque ou déplacement pour pouvoir cliquer plus aisément sur la case voulut.

Lorsqu'une unité attaque, soigne, tue une cible ou autre, un message animé est afficher juste au dessus de cette unité pour indiqué les dégât, le montant du soin ou la mort, l'éboulement d'un bâtiment etc.

L'interface possède plusieurs affichage qui réagissent selon les coordonné de clic de la souris. Nous avons un bouton pour téléporter la caméra sur une unité allié aléatoire, un délais d'attente entre chaque utilisation est par exemple représenté pour une animation du bouton qui reste enfoncé tant que l'action n'est pas disponible à nouveaux.

En haut a droite de l'interface nous possédons un "bash" qui permet d'afficher toutes les informations relative a l'évolution de la grille. Un déplacement de la molette souris permet de scroller vers le haut ou vers le bas. Ces informations possèdent aussi un code couleurs et chaque nouvelle information est recopié dans le fichier de logs.

En haut a gauche s'affiche l'unité ou le bâtiment actuellement sélectionné avec son nombre d'unité tuer, ses statistiques et un code couleur si ces statistique sont modifié par un quelconque bonus ou malus du terrain.

Un système jour nuit est présent avec différente teinte de noire pour la nuit en fonction de l'heure et est synchronisé avec l'animation de l'horloge a droite de l'écran.

7)Optimisations

Pour déterminer les optimisations a effectuer nous avons simulé des partie sur des grille d'environ 20 000 cases avec plus de 5000 joueurs (environ 30 minutes de jeu entre bot a vitesse maximal) pour trouver les problèmes et améliorer l'I.A. .

Une des optimisations à été de réduire la surface de la grille que nous parcourons dans certaine fonction pour se limiter aux cases réellement importante et influente sur le choix et l'action de l'I.A. .

Nous avons aussi limité l'affichage des cases et autres images pour que seule les images présente dans les coordonnée de la fenêtre soient affiché et bien que les autres ne soient pas afficher elles continuent d'exister et évoluer.

Résultats :

La version actuel du projet comporte une grille de jeu limité a 150x150 cases avec un nombre de joueur potentiels de plus de 10000 (bot vs bot ou bot vs humain ou bot vs bot vs humain etc.).

Le jeu est jouable par plusieurs joueur sur la même machine. La grande partie des options du jeu sont modifiable grâce aux quatre menu. Il est possible de sélectionner des unité avant le début de la partie pour certain joueur.

Il est possible de perdre ou de gagner, chaque déroulement de partie est sauvegarder dans un fichier logs unique et il est possible de sauvegarder une partie.

Dans le jeu une interface graphique est présente avec de nombreuses interactions, un bash pour voir les actions de façons précise, un système jour nuit fonctionnel ainsi qu'une horloge indiquant l'avancement dans la journée. Il existe une version vu du dessus et 3D isométrique de l'affichage.

Il est possible a tout moment de mettre le jeu en pose et de le quitter.

Un panel d'animation est disponible pour chacune des unités et quelques images de l'interface.

Une I.A. gère les personnages non joueurs et chaque joueurs peut attaquer, se déplacer, soigner et poser des bâtiments/blocs.

Nous avons aussi une caméra qui peut se déplacer avec la souris ou les touches clavier pour parcourir la grille de jeu.