RAPPORT DE PROJET

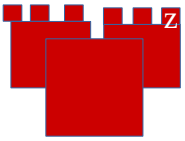
<<les barons>>

SOMMAIRE

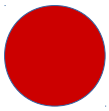
Explication du jeu :

On a 2 équipes, les rouges et les bleus:
(pardonnez moi pour le design, je n'est pas eu le temps de m'attarder dessus)

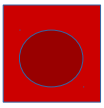
icones rouges:



chateaux



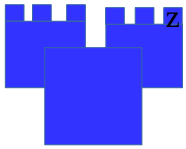
Guerrier



Manants

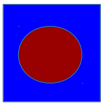
Barons

icones bleus:

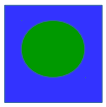


chateaux

Guerrier

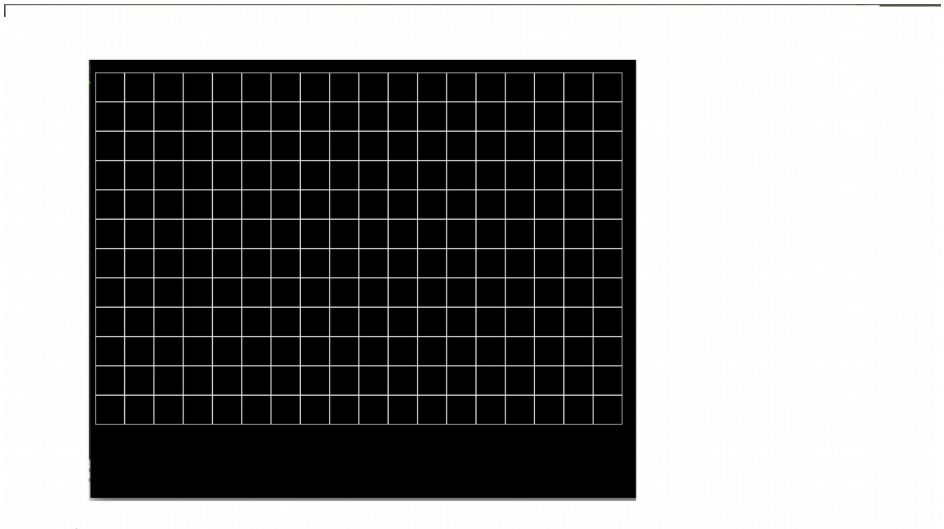


Manants



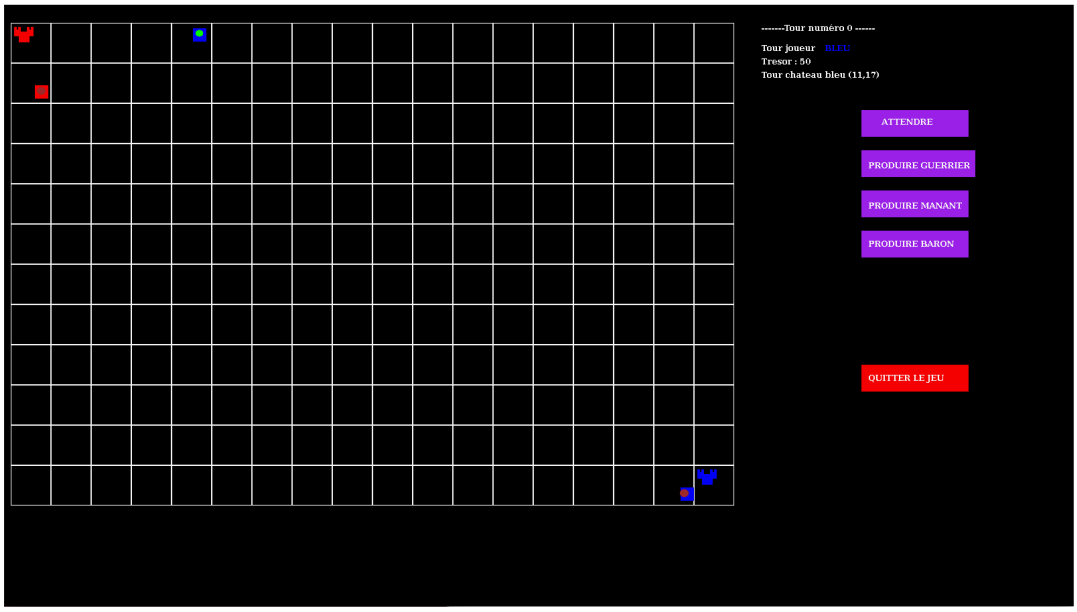
Barons

Et on a le plateau de jeu:

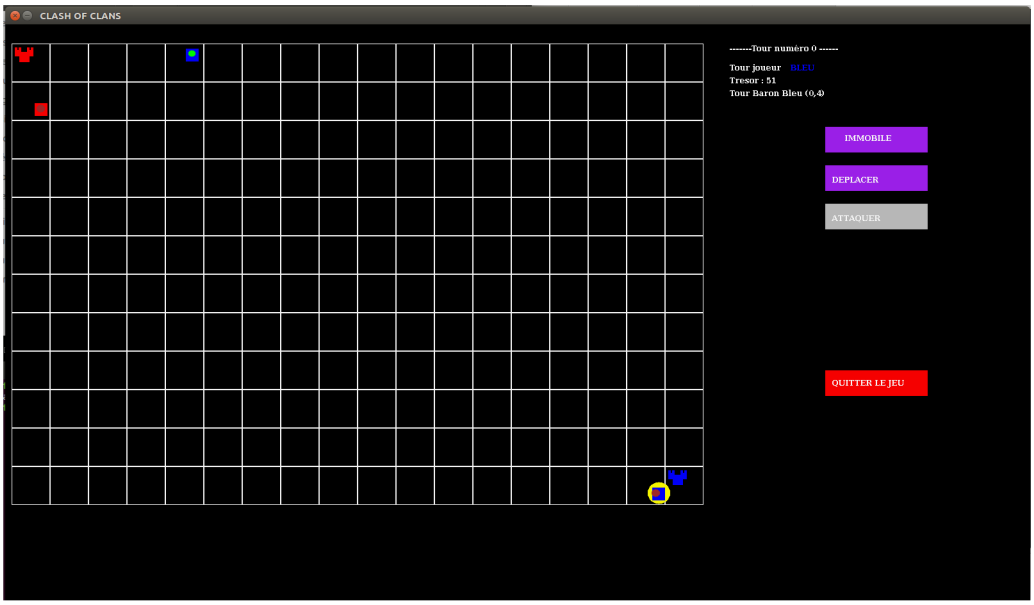


En fonction de l’agent que vous jouez vous aurez des ordres différents:

Exemple avec le château :



Exemple avec le Baron :



Vous avez aussi la possibilité de sauvegarder votre partie.

Je vous laisse découvrir le reste par vous même.
A vos clavier.

2 Les Fonctions:

1.Les choix et complexités:

(elles seront données dans l'ordre dans lequel elles sont écrites dans le programme je ne mettrait que la tête et pas le corps de fonction a cause de problème de taille)

* AListe creation(char genre,char couleur,int posx,int posy)

(fonction permettant d'allouer de la mémoire pour créer un agent ou un château)

Cette fonction est de complexité O(1), car on ne rentre qu'une fois. Aucune boucle, aucune répétition.
Choix le plus simple et le plus logique.

* int execution(Monde *world,char produit,Agent *liste,int tresor,char couleur,int posx,int posy)

(fonction vérifiant les éléments sur les cases alentours du château);
Fonction de complexité O(8) car on ne vérifie que 8 case au pire sinon 1 seule donc O(1).
Il y avait peut être un choix plus logique mais c'est celui qui paraît le plus logique.

* int listeDeplacement(Agent *liste)

(fonction calculant les position choisit par rapport au clic de l'utilisateur)
Fonction de complexité O(n/2) au pire car on calcule la position du clic par rapport a la première case et on déplace de 2 en 2.
Au mieux on a O(1).
Il y avait peut être un choix plus logique mais c'est celui qui paraît le plus logique.

* int detectionEnnemiProche(Agent *tmp,Agent *cible){

(fonction calculant la position des ennemis sur la case du Guerrier/Baron que l'on joue)
Fonction de complexité O(n) car on parcours toute la liste chaînée des ennemis pour vérifier leurs position.

* int detection(Agent *tmp,Agent *cible)

(fonction calculant la position des ennemis sur la trajectoire du Guerrier/Manant que l'on déplace)
Fonction de complexité O(n) car on parcours toute la liste chaînée des ennemis pour vérifier leurs position (comme la fonction précédente).

* int seDeplace(Monde *world,Agent *tmp,Agent *cible,Agent *camp,int siAttaque)

(fonction qui gère le déplacement de nos agents)

Fonction de complexité O(1) car on ne vérifie que une fois les position par rapport à la destination

* int gestionAttaque(Monde *world,Agent *tmp,Agent *cible,Agent *pointe)

(fonction qui gère l'attaque et les éléments à détruire au combat)

Clairement une fonction de complexité $O(n^2)$ du au nombre colossal de déplacement,décalage, vérification et comparaison que l'on fait dans cette fonction.

*int ManantSurCase(Agent *tmp,Agent *cible)

(fonction qui vérifie qu'il n'y ait pas d'autre manant sur la case pour produire les richesse)

Fonction de complexité $O(n^2)$ car on parcours la totalité de la liste chainée pour ensuite la comparer à celle du manant concerné.

*void sauvegarde(Agent *teamRouge, Agent *teamBleu)

(fonction qui permet de sauvegarder la partie)

Fonction de complexité $O(n^2)$.

On parcours la totalité des 2 listes chaînées poour receuillir ls informations nécessaire à la sauvegarde

*int listeMenu(Agent *liste,Agent *cible)

(fonction qui liste les option disponible du menu de chaque agent)

C'est une fonction de complexité O(1)

*void dessineTableau(Agent *rouge,Agent *bleu)

Fonction de complexité $O(n^2)$ voir plus car on fait énormént de sortie de liste et des retours récursifs.

*void fabrication(Agent *tmp)

(fonction qui décrémente le temps de production du château de -1)

Complexité O(1) aucune boucle ni vérification.

*jeu(Monde *world)

(fonction principal du programme)

Complexité $O(n^2)$ très clairement.

*chargerPartie()

(qui affiche l'option charger la partie ou non)

Complexité O(1)

2.2 Principaux problèmes rencontrés:

- Plus gros problème rencontré avec la gestion de pointeur. Je ne pouvais pas supprimer totalement de liste (je pouvais supprimer les agents mais pas le chateaux) chainée hors de la fonction Jeu. C'est pour cela que je le fait directement depuis la fonction jeu.
  Gros problèmes avec la gestion de plusieurs agents sur la même case.

- Problèmes pour gérer l'attaque car il faut savoir si il faut vérifier la case avant ou arrière du baron ou du guerrier.