

# GAME OF STOOLS

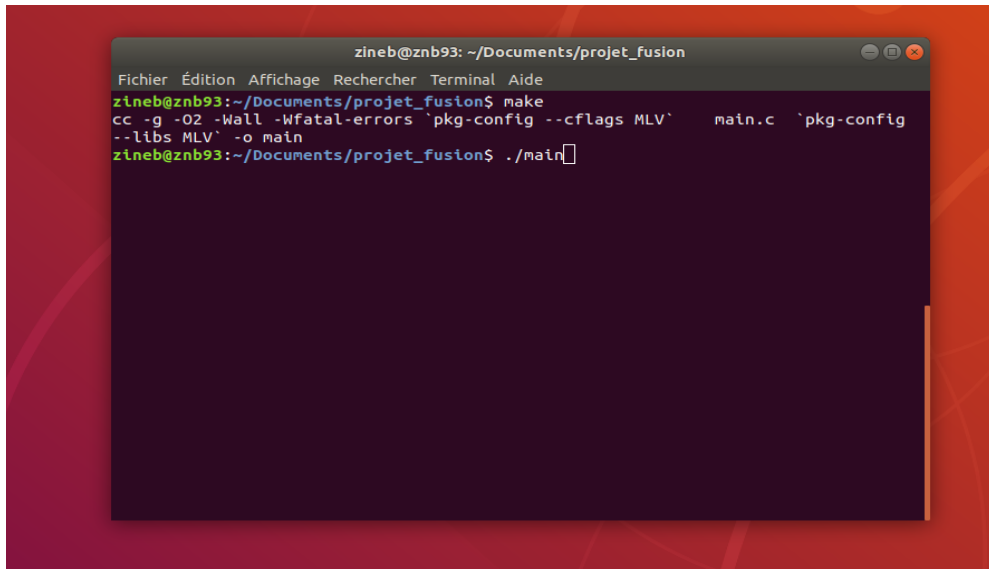
Rapport de projet

## Tables des matières :

- Mode d'emplois
- Organisation du programme ;
- Difficultés rencontrés ;
- Organisation / répartition du travail.

## Mode d'emplois

Pour pouvoir lancer le jeu il faut exécuter la commande [make] et ensuite lancer le fichier ./main.c

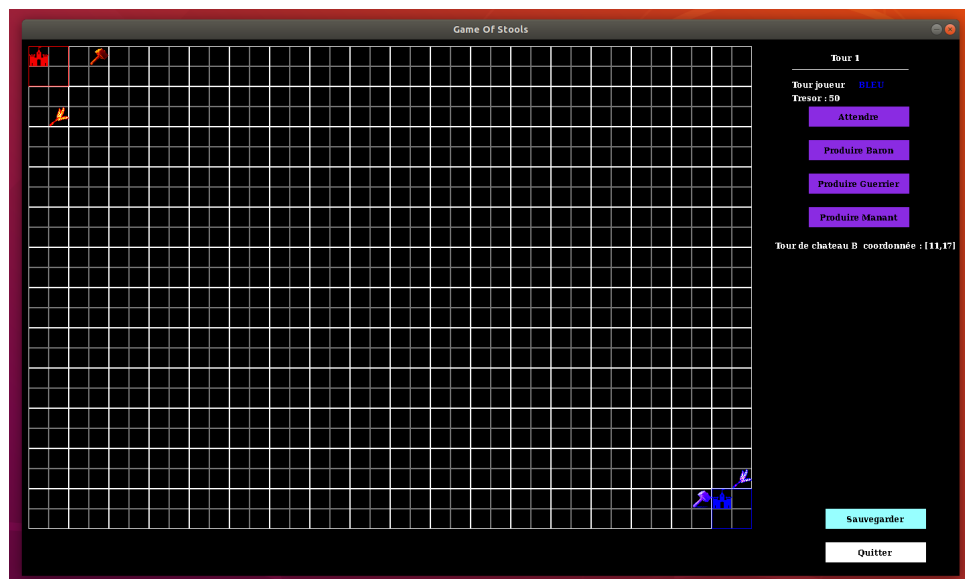


```
zineb@znb93: ~/Documents/projet_fusion
Fichier Édition Affichage Rechercher Terminal Aide
zineb@znb93:~/Documents/projet_fusion$ make
cc -g -O2 -Wall -Wfatal-errors `pkg-config --cflags MLV` main.c `pkg-config --libs MLV` -o main
zineb@znb93:~/Documents/projet_fusion$ ./main
```

À partir d'ici vous aurez le choix entre commencer une nouvelle partie, reprendre une partie déjà sauvegardé ou quitter.



Ensuite vous allez avoir votre grille de jeu contenant l'équipe bleu et rouge.



On vous demande de choisir à chaque fois une action pour le château puis pour ses agents triés par ordre alphabétique pour une même équipe. Le jeu s'arrêtera lorsqu'il n'y aura plus de château dans l'une des deux équipes.

Le château peut passer son tour ou produire des agents, il ne peut pas produire autre choses avant la fin du temps de production de la production en cours.

Chaque agent on la possibilité de rester immobile ou se déplacer dans une case valide. Cependant en fonction de leurs genre ils peuvent obtenir d'autres choix:

- construire un château ou attaquer pour un baron ;
- se transformer en guerrier pour un manant ;
- attaquer et revendiquer ( quand il reste immobile un tour ) pour un guerrier.

Seul les agents sont en mouvement, donc si vous cliquez sur la case déplacer, vous assignez une destination à l'agent en question. Aucune action ne sera possible tant que l'agent ne sera pas arrivé à sa destination.

Lorsqu'un agent de l'équipe adverse se trouve sur la case choisi pour votre agent et que votre est soit un guerrier ou un baron alors un combat à lieu.



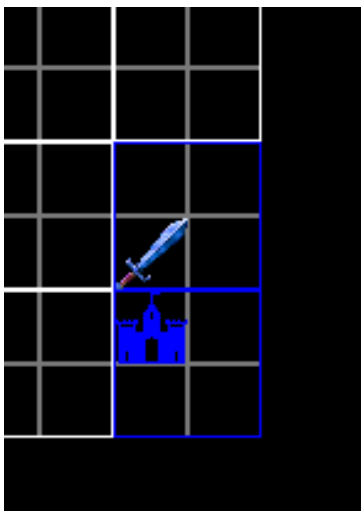
Tour d'un château rouge



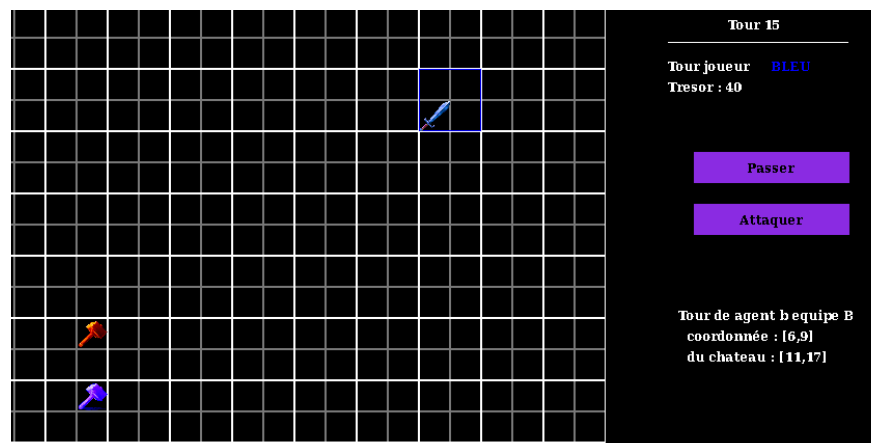
Tour d'un baron



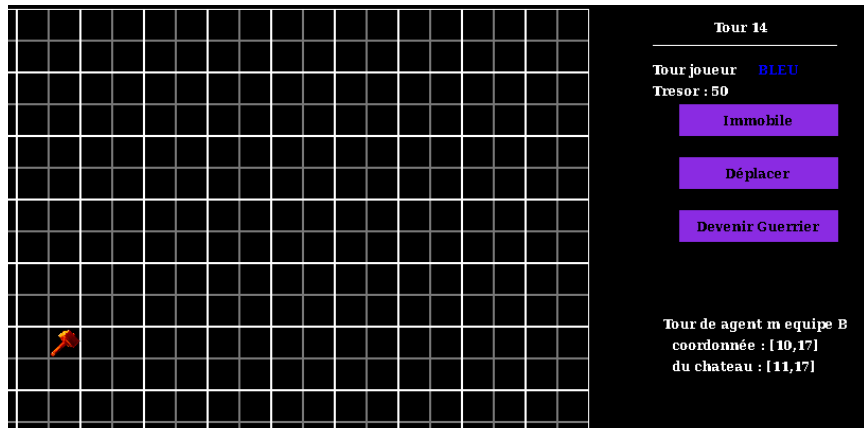
Tour d'un château déjà en production



Case revendiquer par guerrier



Demande d'attaque pour le baron bleu



Suppression de l'agent perdant (bleu), on passe à l'agent suivant

## Organisation du programme

Ses fonctions clefs qui servent à un bon fonctionnement du projet sont listées et décrites ci-dessous :

```
void parcoursClan(char couleur, AListe equipe, Monde *world, int *tresor){
```

```
/*  
-- Cette fonction parcours la liste du clan ("rouge"/"bleu") où est situé notre chateau de  
base  
Elle prend en paramètre la couleur de clan qui joue la liste equipe qui correspond a notre  
liste clan, notre monde en pointeur car on le modifie et enfin le tresor du clan en  
pointeur aussi pour pouvoir le modifier.  
*/
```

```
void parcoursCastle(char couleur, AListe agent, Monde *world, int *tresor, AListe equipe){
```

```
/*  
-- Cette fonction permet de lancer les différentes actions de chaque agent obtenu grâce  
à la liste des habitants d'un même château récupéré dans la fonction parcoursClan  
*/
```

```
void castleAgent(char couleur, char *genre, Monde *world, AListe* chateau){
```

```
/*  
-- On vérifie les positions libre autour du château, ensuite,  
on prend les coordonnées de la position valide qu'on  
affecte à l'agent que l'on veut créer. Ce même agent est  
ajouté dans la liste des habitants du même chateau  
( ie : tous les agents que le château a produit).  
Pour finir on dessine notre monde.  
*/
```

```
void save(AListe clanRouge, AListe clanBleu, Monde world){
```

```
/*
```

```
-- Cette fonction sauvegarde notre partie au point où on l'a laissé
```

```
*/
```

```
void loadedGame(Monde *world){
```

```
/*
```

```
-- Fonction pour commencer avec une partie déjà sauvegarder, on  
récupère pour chaque agent son clan, son genre, sa position etc,  
dans notre fichier de sauvegarde "save.txt". On reprend la  
partie au tour du château qui jouait.
```

```
*/
```

```
int addAndTri(Agent *agent, AListe *listHab, Monde *world, char couleur, char genre, int x,  
int y){
```

```
/*
```

```
-- Cette fonction fait une insertion triée de l'agent que l'on veut  
mettre dans la liste des habitants du château concerné.
```

```
*/
```

```
void attack(Agent *agent1, Agent *agent2, Monde *world){
```

```
/*
```

```
-- On prend les deux agents en combats, en regardant si le genre de ces agents  
respectent les conditions données pour pouvoir lancer le dé et trouver le  
vainqueur pour pouvoir supprimer le perdant du monde. Si le perdant a pour genre  
CHATEAU alors tous ses agents disparaissent sauf les manants que l'on affecte  
au château du gagnant.
```

```
*/
```



## **Difficultés rencontrés**

Nous avons eu du mal à commencer notre projet, du à la compréhension du sujet mais après compréhension de cela, nous avons vite avancer.

Les structure étaient toutes liées entre elles donc on ne savait pas par où commencer ce qui nous a beaucoup ralenti.

Nous avons avancé avec de mauvaise compréhension concernant le sujet donc après plusieurs lignes de codage nous nous sommes rendu compte que nous étions à la limite du hors sujet. Du coup nous avons du réadapter tout notre programme.

On a vu cette erreur grâce à la fonction parcoursClan car dans celle ci on ne parcourait pas les châteaux et ensuite leurs listes d'habitant. On parcourais seulement une seule liste qui contenait le château et les agents en même temps (on avait aucun liens entre le château et ses agents), ce qui causait problème notamment lorsqu'on essayait de trier notre liste seulement d'agent.

## **Organisation / répartition du travail**

Pour la répartition du travail nous avons décidé de travailler ensemble sur chaque point du projet pour pouvoir avancer plus vite sur le travail à fournir. Avec l'aide GitHub nous avons toujours à porter de main les archives de notre projet .

De plus, nous mettons nos programmes à exécution pour aider à corriger un bug et/ou une erreur non vu par celle qui a écrit le programme.