# Projet de programmation – L2 2018-2019

**Game of Stools**

**Abstract**

Le clan bleu et le clan rouge se disputent un territoire.
Le but du projet est de simuler cet affrontement à l'aide de la bibliothèque graphique de l'université MLV.
Le projet est à développer par niveaux de difficultés successifs.

Deux groupes (Bleu et Rouge) sont présents.
Il y a quatres types d'éléments dans chaque groupe:

- les châteaux qui ne se déplacent pas, et qui peuvent produire des barons, des guerriers, ou des manants;
- les barons, qui se déplacent, attaquent ou construisent un château;
- les guerriers, qui se déplacent et attaquent;
- les manants, qui se déplacent ou produisent des richesses. Un manant ne peut pas attaquer, mais peut prendre les armes en se transformant définitivement en guerrier.

Les cases peuvent être rouge, bleu ou neutre.
Dans la suite on appellera *agents* les éléments mobiles. Un agent se déplace d'au plus une case, vers une case voisine.

1. **Mise en place**
   À chaque tour, on tire au hasard le groupe qui agira en premier.
   On traite les actions de tous les éléments d'un groupe, puis tous ceux de l'autre groupe.
   Les productions des châteaux sont traitées, puis le déplacement des combattants, puis la production ou le déplacement des manants.
   Lorsqu'un déplacement entraîne la rencontre d'agents de couleurs différentes, alors un combat a lieu (voir section Attaque).
   Initialement, il y a un château, un baron et un manant pour chaque couleur, placés dans des coins opposés. Toutes les cases sont neutres.
   Le trésor initial de chaque clan est fixé à 50.

2. **Production**

   - Les châteaux
     Lorsque la production d'un château a été choisie par le joueur, elle ne peut plus être changée avant sa réalisation complète.
     Après réalisation d'une production, le château attend un nouvel ordre de production. On peut choisir de produire (valeurs par défaut):
     - un baron (6 tours, coût 20);
     - un manant (2 tours, coût 1).

     Chaque château est relié à tous les agents qu'il a produit. Si le château est détruit, tous les barons et guerriers qu'il a produit disparaissent immédiatement et les manants changent leur allégeance (ils sont alors reliés au même château que l'*agent* qui a détruit leur ancien château).
     Lorsqu'un baron construit un château, il se rattache alors à ce château.
   - Les manants
     - Lorsqu'il est immobilisé sur une case de la couleur de son clan, un manant produit des richesses: le trésor de son clan augmente de 1 à chaque tour.
     - Moyennant un coût (par défaut 5) un manant mobile peut se transformer en guerrier. La transformation est immédiate, sans déplacement. Le guerrier pourra se déplacer au tour suivant.

3. **Revendiquer une case**
   Seuls les guerriers peuvent revendiquer une case. Il suffit qu'un guerrier reste immobile pour que la case prenne immédiatement sa couleur. Elle gardera cette couleur jusqu'à une nouvelle revendication.

4. **Déplacement**
   Les châteaux ne se déplacent pas et deux châteaux ne peuvent pas se trouver sur une même case.
   Chaque agent a une destination.
   Si sa position n'est pas égale à sa destination, il se rapproche d'une case de sa destination.
   Si sa position est égale à sa destination, il attend un nouvel ordre qui peut être :

   - la destruction de l'agent;
   - une nouvelle destination (en cliquant sur le plateau);
   - le sur-place:
     lorsqu'il indique comme nouvelle destination la case où il se trouve déjà l'agent reste sur la case pour au moins un tour.
     Si l'agent est un manant, un choix est proposé:
     - prendre les armes et se transformer en guerrier qui pourra bouger au tour suivant;
     - récolter des ressources, si la case est de la couleur de son clan. Il devient alors immobile et le reste jusqu'à la fin de la partie ou jusqu'à être détruit ou à changer d'allégeance. Il ne peut y avoir au plus qu'un seul manant immobile par case (il faut voir chaque case comme un champ de ressources);

– ne rien faire (la case n'est pas de la bonne couleur ou est déjà exploitée).

Un guerrier restant sur sa case revendique cette case qui prend immédiatement la couleur de son clan.
Un baron restant sur sa case peut construire un château s'il n'est pas déjà sur un château et que le trésor est suffisant. Le coût de la construction d'un château est 30.

Pour repérer aisément un agent en attente d'ordre, on affiche un message en mode terminal(genre, coordonnées) on le fait resortir par une particularité en mode graphique (graphisme, couleur, clignotement).

5. **Attaque**
   Si un agent guerrier (baron ou guerrier) veut se placer sur une case occupée par des éléments de couleur différente, un combat a lieu. Il est résolu par tirage aléatoire d'un dé à 100 faces (fonction `int MLV_integer_random (int begin, int end)` par exemple). Un coefficient égal au coût de production sera appliqué : si un baron et un guerrier combattent, on compare donc `tireDe()`×`CBARON` et `tireDe()`×`CGUERRIER`. A l'issu d'un combat, l'un des éléments est détruit.
   Lorsque plusieurs éléments d'une couleur sont sur une même case, le processus se répète tant qu'une des couleurs n'est pas éliminée. On ordonnera judicieusement les éléments placés sur une même case pour que les guerriers interviennent en premier, puis les barons et enfin les manants. Si un château est présent, il n'est détruit que lorsqu'il se trouve attaqué alors qu'il n'a plus aucune protection.

6. **Le plateau de jeu**
   Le plan est considéré comme un quadrillage de cases carrées, chaque case non adjacente à un bord possède 8 voisins.
   Les bords du plateau ne sont pas franchissables.

7. **Fin de partie**
   Une partie est terminée lorsque l'une des couleurs ne possède plus de château. On peut également choisir d'arrêter le jeu.

8. **Sauvegarde et chargement**
   Au début du tour, après le tirage aléatoire, le clan jouant en premier peut choisir de sauvegarder la partie dans un fichier. La couleur devant agir, ainsi que tous les éléments, devront être mémorisés (voir format de sauvegarde).
   On devra également pouvoir choisir de récupérer une partie mémorisée dans un fichier.
   À chaque fois, l'utilisateur entrera le nom du fichier de sauvegarde.

## Implantation et niveaux de jeu

Le jeu se joue au tour par tour. À la fin de son tour, le joueur doit indiquer sa *fin de tour* afin de rendre la main à son adversaire.

1. **Interface graphique**
   La fenêtre graphique est séparée en différentes parties qui doivent respecter les impératifs suivants:

   (a) la première partie représente le plateau de jeu, quadrillé. Les cases sont de taille 60 par 60 pixels. Chaque case se découpe en quatre sous-carrés $30 \times 30$ permettant de représenter chaque élément présent;

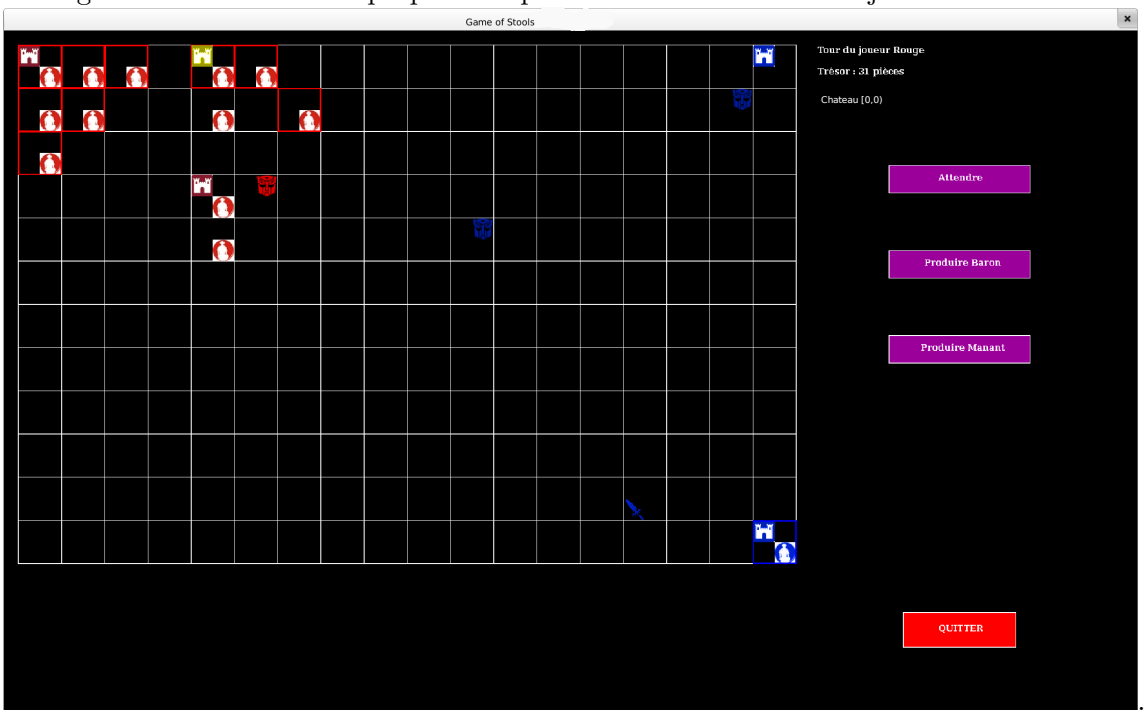
   - un château en haut à gauche;
   - les barons en haut à droite;
   - les guerriers en bas à gauche;
   - les manants en bas à droite.

   On pourra représenter symboliquement le château et les agents ou utiliser des images, `man MLV_image.h`; La couleur de fond de la case permet de savoir si elle a été revendiquée.

   (b) la seconde partie forme une barre de menu contenant des boutons permettant la gestion des différents ordres à transmettre aux manants, guerriers, barons et châteaux; elle comprend également une partie dans laquelle s'afficheront les coordonnées de l'agent courant, l'action qu'il est en train d'effectuer;

   (c) enfin une dernière zone contient un bouton *sauve*, un bouton *charge*, un bouton *quitter*, un bouton *fin de tour* ainsi que l'indication du numéro du tour et la couleur du joueur en cours.

L'image suivante donne une proposition possible d'une situation de jeu.




. L'utilisation

d'une interface graphique remplace celle de l'interface texte, sauf pour le niveau 1.

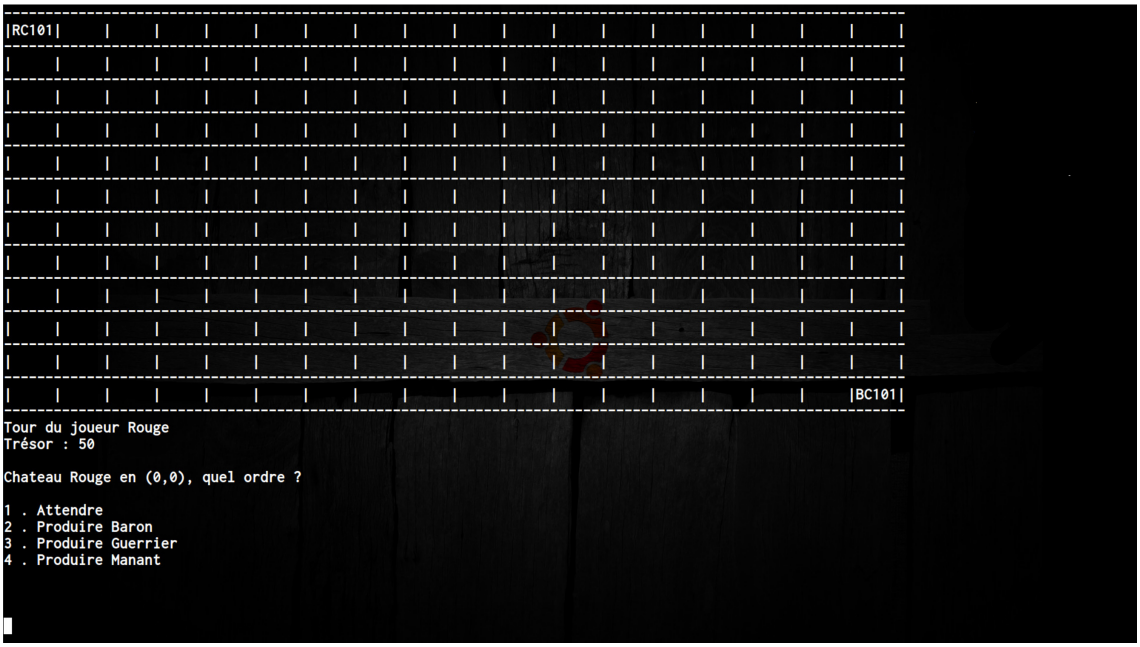
## 2. Interface texte

Le cœur du projet est la manipulation de listes chaînées. Pour que l'interface graphique ne constitue pas un obstacle infranchissable, il vous est possible de commencer par une interface en mode terminal. Il s'agit alors de présenter le plateau à l'aide de caractères ASCII. Le moins (`-`) de votre clavier peut faire des lignes horizontales et le pipe (`|` `AltGr` + 6) peut fabriquer les parois verticales. Ensuite, dans chaque case, 5 caractères désignent le contenu:

- premier caractère : le clan `'R'` pour Rouge et `'B'` pour Bleu;
- second caractère : le caractère `'C'` si un château est présent, le caractère `' '` sinon;
- un caractère `X` : le nombre de barons;
- un caractère `Y` : le nombre de guerriers;
- un caractère `Z` : le nombre de manants.

On supposera qu'il ne peut y avoir plus de 9 unités de même type sur une même case. Ainsi, une case occupée par un château bleu avec 2 barons et 1 guerrier (forcément de même couleur) est représentée par `BC210`. La couleur d'une case est symbolisée par 'r' 'b' ou ' '. Si la case est occupée la couleur n'apparait pas.

Voici alors une image du plateau de départ avec une telle représentation :



3. **Structures et représentations des données**
   Les châteaux, barons, guerriers et manants sont représentés par la même structure `Agent`.
   En fonction des niveaux réalisés, certains champs peuvent ne pas être utilisés.
   On utilisera les constantes symboliques et les types suivants:

```
/* dimension du monde en nombre de cases*/
#define NBCOL 18
#define NBLIG 12
/* l'origine est en haut a gauche*/
/* les deux clans */
#define ROUGE 'R'
#define BLEU  'B'
#define LIBRE '-'
/* les genres d'agents */
#define MANANT 'm'
#define BARON  'b'
#define GUERRIER  'g'
#define CHATEAU 'c'
/* les temps de production */
#define TMANANT 2
#define TGUERRIER 4
#define TBARON 6
/* les couts */
#define CMANANT 1
#define CGUERRIER 5
#define CBARON 10
#define CCHATEAU 30

typedef struct agent{
  char clan; /* ROUGE ou BLEU */
  char genre; /* BARON, MANANT, GUERRIER, CHATEAU */
  char produit; /* production actuelle d'un chateau*/
  int temps; /* tours restant pour cette production */
  int posx, posy; /* position actuelle */
  int destx, desty; /* destination (negatif pour manant immobile) */
  struct agent *asuiv, *aprec; /* liste des agents liees a un chateau */
  struct agent *vsuiv, *vprec; /* liste des voisins */
}Agent,*AListe;

typedef struct{
  Agent *chateau; /* s'il y a un chateau sur la case */
  AListe habitant; /* les autres occupants */
  char clan;/* couleur du clan ayant revendique*/
}Case;

typedef struct{
```

```
  Case plateau[NBLIG][NBCOL];
  AListe rouge, bleu;
  int tour; /* Numero du tour */
  int tresorRouge, tresorBleu;
}Monde;
```

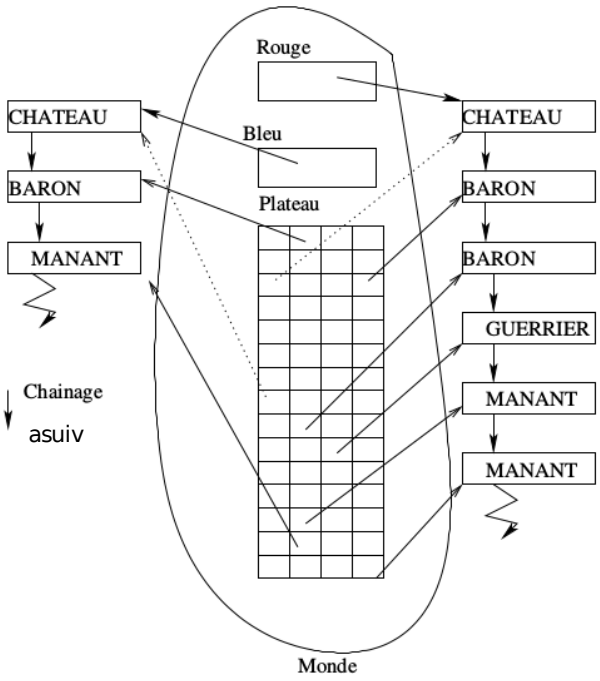
4. **Format de sauvegarde**
   Le format des fichiers de sauvegarde est fixé. Il vous permet de construire des fichiers de tests.
   La première ligne contient un `char` et 3 `int` séparés par un espace: la lettre précisant la couleur du clan devant jouer, le nombre de tours et la valeur du trésor du clan devant jouer puis la valeur du trésor de l'autre camp.
   Sur chacune des lignes suivantes, on trouve les caractéristiques d'un seul élément (sauf bien entendu les valeurs des pointeurs). Une ligne est donc constituée de `3 char` suivis de `5 int` séparés par un seul espace. Les `int` indiquent dans cet ordre le temps de production la position actuelle la destination
   `Rcb 2 3 4 3 4` désigne un chateau (`c`) `rouge (R) qui produira un baron (b` dans 2 tours et qui est situé sur la case (3,4).
   `Bm- 0 4 5 -1 5` désigne un manant bleu immobile situé case (4,5)
   `Bg- 0 8 8 12 7` désigne un guerrier bleu situé case (8,8) qui se dirige case (12,7).

   Un agent est relié au premier château qui le précède.
   La couleur des cases du plateau est ensuite donné directement par une suite de 12 lignes de 18 caractères `'R','B',' '`.

5. **Les niveaux**
   Le projet est à développer par niveaux. Ces niveaux, de difficultés croissantes, correspondent à des notes maximales croissantes.

   (a) Niveau 1
   Dans ce niveau, on demande uniquement de gérer les déplacements des agents et la production d'un unique château. On doit pouvoir choisir la production, déplacer, immobiliser ou détruire un élément.
   Deux éléments ne peuvent pas être placés sur la même case du plateau. Un agent produit par un château est placé sur une case libre voisine de celui-ci. S'il n'y a pas de case voisine libre la production reste en attente qu'une case se libère.
   Chaque clan forme une liste chaînée, grace aux pointeurs `asuiv`, dont le premier élément est le chateau. Une case du plateau de jeu pointe sur l'élément situé sur cette case.

   Pour ce niveau (et lui seul) ilfaut afficher la liste ordonnée (barons, guerriers puis manants) de tous les agents du clan ainsi que la liste ordonnée par ligne,colonne des cases occupées (coordonnées et occupant!)

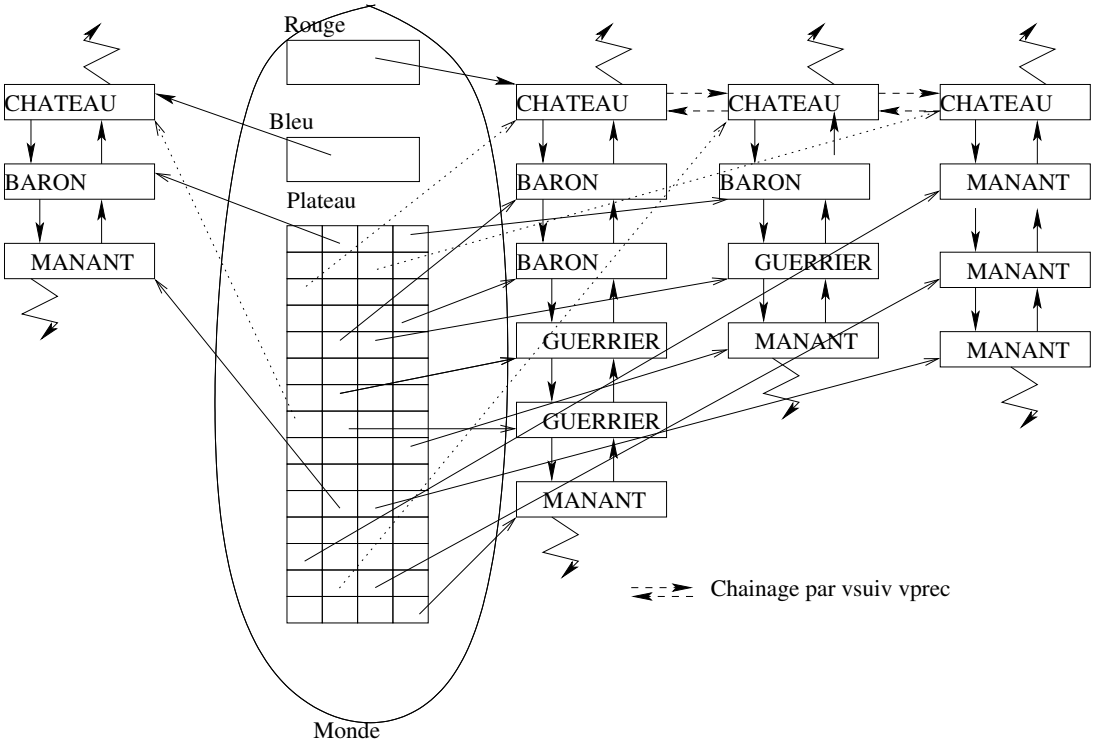



Niveau 1 ,

(b) Niveau 2
En plus du Niveau 1, on ajoute la possibilité pour un manant de prendre les armes en se transformant en guerrier. Le genre de l'agent est changé et le nouveau guerrier est déplacé dans sa liste en dernière position de la partie des guerriers (pas de destruction de cellule de la liste!).
On ajoute dans ce niveau la gestion des sauvegardes et chargements. Le nom de la sauvegarde et du fichier à charger sera entré par l'utilisateur. On demandera tous les 5 tours si une sauvegarde ou un chargement est demandé.

(c) Niveau 3
En plus du Niveau 2, on ajoute la possibilité pour un baron de créer de nouveaux châteaux. Le nouveau château est ajouté à la liste chaînée des châteaux de sa couleur grâce au pointeur `*vsuiv`.
Pour ces trois premiers niveaux de difficulté, on pourra se limiter à des listes simplement chaînées en n'utilisant que les champs des pointeurs `*asuiv` et `*vsuiv`.

(d) Niveau 4
En plus du Niveau 3, on ajoute la gestion des combats entre 2 éléments de couleurs différentes. Pour ce niveau, il faudra utiliser le double chaînage pour l'allégeance et les voisins (`*asuiv`, `*aprec` et `*vsuiv`, `*vprec`).




Niveau 3

(e) Niveau 5
L'utilisation d'une interface graphique devient obligatoire. Les choix de l'utilisateur seront définis par des menus cliquable.

(f) Niveau 6
On autorise désormais la possibilité que plusieurs agents soient placés sur la même case. Pour cela, la liste doublement chaînée des voisins d'un agent relie tous les agents situés sur la même case.

## Améliorations

De nombreuses améliorations sont possibles. Elles ne seront prises en compte que si tous les niveaux précédents sont totalement réalisés.
On peut par exemple:

- ajouter un déplacement en mode patrouille : aller-retour entre un point et le château;
- faire intervenir le nombre d'agents présents sur une même case dans les combats;
- ajouter un système de points de vie et de points d'attaque, châteaux, barons, guerriers ou manants ayant un nombre de points différents;
- relier la survie et la rapidité de production d'un château au nombre de manants qui lui sont attachés.
- ...

## Ce qu'il faut faire

Le projet est à effectuer en binôme du même TP, sauf accord des 2 chargés de TP. Une soutenance aura lieu.

Les programmes C **compilables à l'université** devront être clairs et judicieusement commentés.

Un rapport décrivant d'une part le fonctionnement, d'autre part les méthodes utilisées et les choix d'implantation sera rendu. La complexité des principales fonctions devra être expliquée. Le but est de permettre à un utilisateur quelconque d'utiliser le programme, et à un programmeur averti de faire évoluer facilement votre code.

Ne commencez pas à développer un niveau avant que le niveau précédent soit totalement fonctionnel. Sauvegardez, niveau par niveau, des versions fonctionnelles pour pouvoir repartir sur de bonnes bases lorsque de nouvelles fonctionnalités se révèlent trop difficiles à implanter. Chacune des versions pourra être testée le jour de la soutenance.

La compilation ainsi qu'une démonstration de votre code sur une machine de l'université fait partie de la soutenance.