Projet d'algorithmique – L2 2011-2012

19 mars 2012

Luttes moyenageuses

Résumé

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 chateaux qui produisent des seigneurs, des guerriers, ou des manants et ne se déplacent pas ;
- les seigneurs qui se déplacent, attaquent ou se construisent un chateau et deviennent alors sédentaires
- les guerriers qui se déplacent et attaquent .
- les manants qui se déplacent ou produisent des richesses.

Dans la suite on appellera *agents* les éléments mobiles. Un agent se déplace au plus d'une case, vers une case voisine.

1. **Mise en place**
   Chaque couleur agit alternativement. A chaque tour, on traite tous les éléments d'un groupe puis tous ceux de l'autre groupe.
   Les productions des chateaux sont traitées, puis la production ou le déplacement, d'au plus une case, de chaque agent.
   Initialement, il y a un chateau, un seigneur et un manant pour chaque couleur, placés dans des coins opposés. Le trésor de chaque clan est fixé à 50.
2. **Production**
   La production d'un chateau est choisie et ne peut être changée avant sa réalisation complète. On peut choisir de produire :
   – un seigneur (6 tours coût 20),

– un guerrier (4 tours coût 5)
– un manant (2 tours coût 1).

Après réalisation d'une production, la chateau attend un nouvel ordre de production.

Chaque chateau est relié à tous les agents qu'il a produit. Si le chateau est détruit tous les seigneurs et guerriers qu'il a produit disparaissent immédiatement et les manants changent leur allègeance (ils sont alors reliés au même chateau que celui qui a détruit leur ancien chateau) .

Si un seigneur se transforme en chateau, il n'est plus attaché au chateau qui l'a produit.

3. **Déplacement**

   Les chateaux ne se déplacent pas et deux chateaux 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.

   Cet ordre peut être :

   – la destruction de l'agent.
   – une nouvelle destination (en cliquant sur le plateau).
   – immobilité.

     Un manant immobile le reste jusqu'à la fin de la partie ou jusqu'à être détruit ou changer d'allégeance. A chaque tour, un manant immobile ajoute 1 au trésor de son clan. Un manant redevient mobile lorsqu'il change d'allégeance.

     Un seigneur immobile se transforme en chateau s'il n'est pas déjà sur un chateau et que le trésor est suffisant. Le coût de la transformation en chateau est 30.

   Pour repérer aisément un agent en attente d'ordre on l'affiche avec une couleur particulière.

4. **Attaque**

   Si un agent guerrier (seigneur 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 (fonction `int MLV_integer_random (int begin, int end)` par exemple). Un coefficient égal au coût de production sera appliqué. A l'issu d'un combat, l'un des éléments est détruit. Lorsque plusieurs éléments d'une couleur peuvent être 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 seigneurs et enfin les manants. Si un chateau est présent,il n'intervient que dans le dernier combat.

5. **Le plateau de jeu**
   Le plan est considéré comme un quadriage de cases carrées, chaque case possède 8 voisins.
   Les bords du plateau ne sont pas franchissables.
6. **Fin de partie**
   Une partie est terminée lorsque l'une des couleurs ne possède plus de chateau. On peut également choisir d'arrêter le jeu.
7. **Sauvegarde et chargement**
   Au début du tour de chaque clan, on peut choisir de sauvegarder la partie dans un fichier. La couleur devant agir ainsi que tous les éléments devront être mémorisés.
   On devra de même pouvoir choisir de récupérer un partie mémorisée dans un fichier.
   L'utilisateur entrera le nom du fichier de sauvegarde.
8. **Issues des combats choisis**
   Un mode de contrôle permettant de choisir les résultats des combats devra être disponible.

Implantation et niveaux de jeu

1. **Interface graphique**
   L'interface graphique doit respecter les impératifs suivants :
   - La fenêtre graphique est séparée en deux parties.
   - La partie supérieure représente le plateau de jeu. Les éléments occupent des carrés de 10×10 pixels. On pourra représenter un chateau par un carré vide et les agents par des carrés pleins de couleurs différentes.
     Le quadrillage du plateau de jeu est apparent.
   - La partie inférieure forme une barre de menu contenant des boutons permettant la gestion des différents ordres à transmettre aux manants, guerriers, seigneurs et chateaux ; elle comprend également une partie dans laquelle s'afficheront les coordonnées de l'agent courant, l'action qu'il est en train d'effectuer. Enfin il doit y avoir une troisième zone dans laquelle se trouve un bouton *sauve*, un bouton *charge*, un bouton *quitter*, un bouton *fin de tour* ainsi que l'indication du

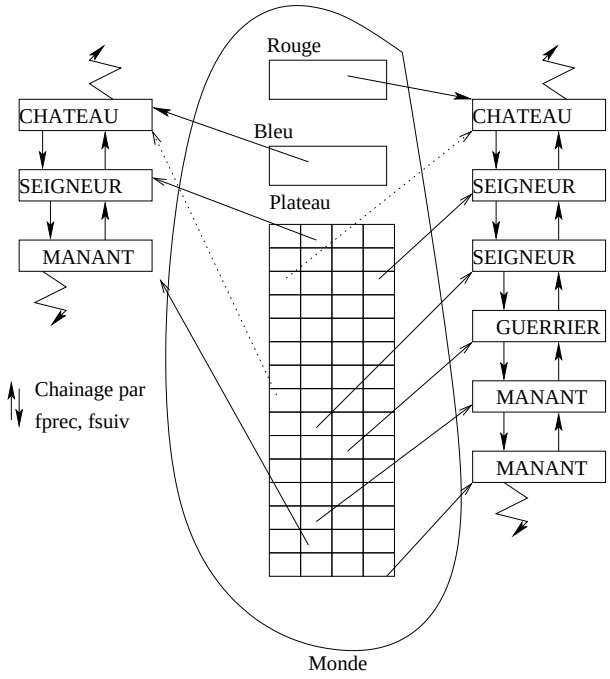
numéro du tour et la couleur du joueur en cours.
Le jeu se joue au tour par tour. A la fin de son tour le joueur doit cliquer sur le bouton *fin de tour* afin de rendre la main à son adversaire.

2. **Structures et représentations des données**
Les chateaux, seigneurs, guerriers et manants sont représentés par la même structure `Personnage`.
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 X 60
#define Y 40
/* L'origine est en haut \'a gauche*/
/*les deux clans*/
#define ROUGE 'R'
#define BLEU  'B'
/*les genres d' agents*/
#define MANANT 'm'
#define SEIGNEUR  's'
#define GUERRIER  'g'
#define CHATEAU 'c'
/*les temps de production*/
#define TMANANT 2
#define TGUERRIER 4
#define TSEIGNEUR 6
/*les couts*/
#define CMANANT 1
#define CGUERRIER 5
#define CSEIGNEUR 10
#define CCHATEAU 30
typedef struct four{
char couleur;/* ROUGE ou BlEU*/
char genre;/*SEIGNEUR,MANANT,GUERRIER,CHATEAU */
int posx,posy;/*position actuelle*/
int destx,desty;/*destination (negatif si immobile)*/
int produit,temps;/*production actuelle d'un chateau*/
  struct four *fsuiv,*fprec;/*liste des agents liees a un chateau*/
  struct four *vsuiv,*vprec;/*liste des voisins*/
```

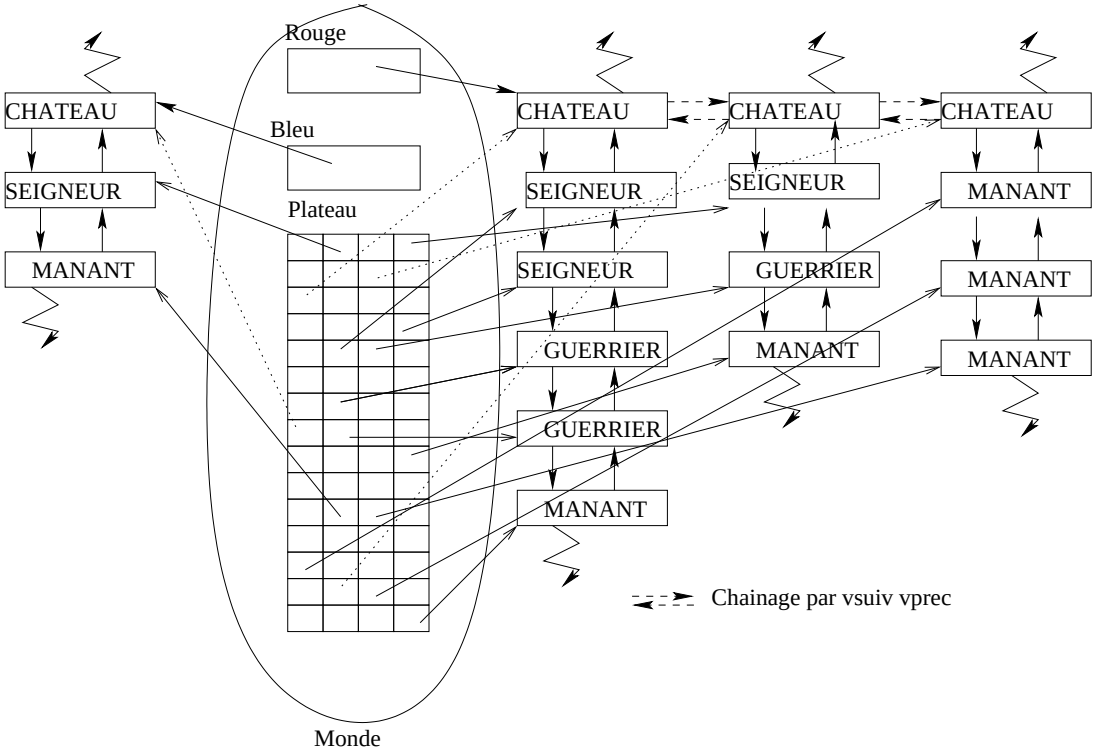
```
}Agent,*AListe;

typedef struct
{Agent * Chateau;/* s'il y a un chateau sur le case*/
  AListe  habitant;/* les autres occupants*/
}Case;
typedef struct
{Case plateau[X][Y];
  FListe Rouge,Noir;
}Monde;
```

3. **Format de sauvegarde**
   Le format des fichiers de sauvegarde est fixé.
   La première ligne contient la lettre précisant la couleur du groupe devant jouer et la valeur de son trésor. 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 `2 char` suivis de `6 int` séparés par un seul espace.
   Un agent est relié au premièr chateau qui le précède.
   Ce formatage doit permettre d'échanger des fichiers de tests.
4. **Les niveaux** Le projet est à développer par niveaux. Ces niveaux, de difficulté 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 chateau. 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 chateau est placé sur une case libre voisine de celui-ci.
   Afin de pouvoir aisément détruire un élément, les éléments d'une couleur sont reliés par une liste doublement chainée. Une case du plateau de jeu pointe sur l'élément situé sur cette case.



Niveau 1 ,

(b) Niveau 2
En plus du Niveau 1 on ajoute la possibilité pour un seigneur de se transformer en un nouveau chateau. Un seigneur qui se transforme en chateau se retire de la liste des agents. Le chateau est ajoutée à la liste doublement chaînée des chateaux de sa couleur grâce aux pointeurs `*vsuiv,*vprec`.



Niveau 2

,

(c) Niveau 3
En plus du Niveau 2 on ajoute la gestion des combats entre 2 éléments de couleurs différentes.

(d) Niveau 4
On autorise désormais la possibilité que plusieurs agents soit 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 chateau.
– Ajouter un système de points de vie et de points d'attaque, chateaux, seigneurs, guerrier ou manants ayant un nombre de points différent.
– Relier la survie et la rapidité de production d'une chateau au nombre de manants qui lui sont attachées.
– ...

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 pendant le dernier TP.
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 commencer pas à développer un niveau avant que le niveau précédent soit totalement fonctionnel.