Présentation du jeu Ataxx

Olivier Marchetti.
Polytech-Paris Sorbonne

1 Votre travail pour cette séance

Il vous est demandé pour cette séance de :

1. lire attentivement le sujet (vous pouvez nous poser des questions) ;
2. réfléchir à l'ergonomie et notamment au « *gameplay* » ;
3. proposer une modélisation la plus détaillée possible et de façon lisible (fichier informatique ou *scan* de votre travail). Les différents identifiants de vos abstractions devront être nommés judicieusement ;
4. réfléchir aux éventuelles structures de données utiles à vos classes.

Nous relèverons en fin de séance ce travail. Vous aurez courant janvier d'autres instructions (notamment concernant le rendu final). Enfin, pour démarrer votre mini-projet, un fichier permettant la colorisation d'un terminal classique sous Linux vous sera donné.

Attention : ce travail devra être une création originale et personnelle. Tout emprunt devra être dûment sourcé et notifié (dans le cas contraire, les règlements en vigueur assimilent ce type de situation à un cas de triche). En cas de copie, une note sanction sera appliquée indisctinctement à toutes les personnes (dont l'auteur de la source).

2 Introduction

Le jeu Ataxx est un jeu de stratégie créé par Dave Crummack et Craig Galley en 1988 pouvant être joué par deux joueurs. Le jeu se compose :

— d'un plateau de quarante neuf cases disposées en carré ;
— de quatre pièces de couleur rouge ou bleue (en début de jeu) ;
— de quatre cases comportant des obstacles inamovibles.

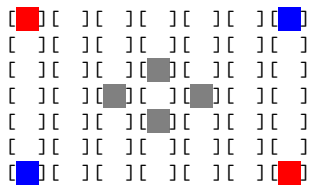
Vous trouverez une démonstration du jeu sur l'excellent site internet `archive.org` :

https://archive.org/details/arcade_ataxx#

Objectif : réaliser une version graphique en mode texte de ce jeu (version sans limite de temps pour jouer).

3 Description du jeu

En début de partie, les pièces d'une même couleur sont disposées sur deux coins opposés du plateau (l'une des pièces rouges occupant dans le coin supérieur gauche) comme indiqué sur cette figure (les obstacles étant en gris) :



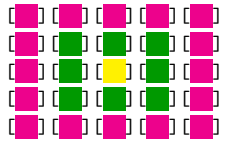
Chaque joueur dispose initialement de deux pièces de même couleur (le joueur ayant les pièces rouges commence la partie) et les joue chacun son tour.
Lorsque vient le tour d'un joueur, celui-ci doit choisir une case hébergeant l'une de ses pièces puis décider d'un coup vers une case vide se trouvant dans l'un des deux voisinages autorisés. Nous introduisons les définitions formelles des deux types de voisinage possible.

Voisinage immédiat : une case se trouve dans ce voisinage si elle
— elle est vide ;
— adjacente à la case de départ et vide (donc huit cas possibles au maximum selon la configuration du jeu) ;

Voisinage distant : une case se trouve dans ce voisinage si
— elle est vide ;
— et de plus,
  — elle est distincte de toute case de son voisinage immédiat ;
  — elle est se trouve dans l'un des voisinages immédiats de l'une des case du voisinage immédiat de la case initiale.

Si ces deux ensembles sont vides (toutes ces cases étant occupées par des pièces ou des obstacles), alors le joueur passe obligatoirement son tour.

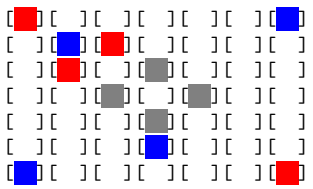
Sur le schéma suivant, si l'on considère un plateau sans aucun obstacle, la case jaune représente la pièce que la joueur à décider de jouer, tandis que les cases vertes (*resp.* magenta) représentent le voisinage immédiat (*resp.* distant).



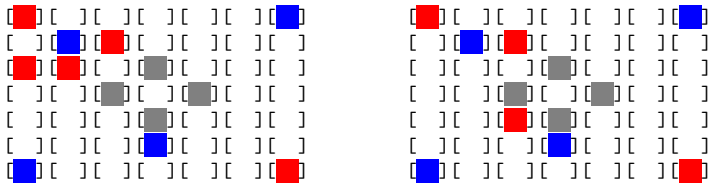
Possibles effets d'un mouvement sur les pièces du joueur :

— si le joueur décide de jouer un coup dans son voisinage immédiat, la pièce initiale n'est pas déplacée et une nouvelle pièce de même couleur est ajoutée dans la case ciblée ;
— si le joueur décide de jouer un coup dans son voisinage distant, alors la pièce est déplacée de sa case initiale vers la case ciblée.

À titre d'exemple, supposons que l'état du jeu soit caractérisé par le plateau suivant et que le joueur courant dispose des rouges et décide de déplacer la pièce rouge en troisième ligne et deuxième colonne :



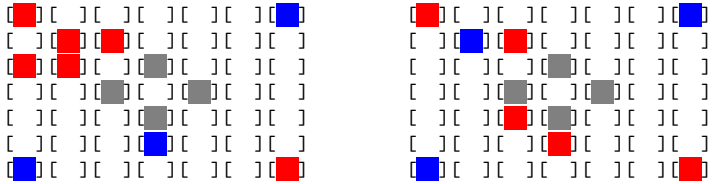
Voici les deux configurations obtenues selon que le joueur décide de déplacer sa pièce vers la case adjacente gauche (figure de gauche), ou vers la case en cinquième ligne et troisième colonne (figure droite) :



Dans les deux cas, vous noterez qu'une unique case vide du jeu se trouve modifiée et ainsi remplacée par une pièce du joueur.

Possibles effets d'un mouvement sur les pièces de l'adversaire : à l'issue d'un mouvement, toutes les pièces adverses se trouvant dans le voisinage immédiat de l'unique case modifiée par le mouvement seront « infectées ». L'infection d'une pièce consistera en la modification de sa couleur par la couleur de l'adversaire ayant joué le coup.

Voici les deux configurations obtenues après infections pour les deux configurations précédemment illustrées.



Dans les deux cas, une pièce adverse aura été infectée et sera devenue rouge (ce même nombre de pièces infectées est ici purement fortuit).

Fin du jeu : la partie s'arrête dès que l'un des joueurs ne peut plus jouer, soit par une absence totale :

— de pièces de sa couleur ;

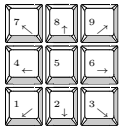
— de case vide (le plateau étant plein) ;

. Le vainqueur sera alors le joueur ayant le plus de pièces sur le plateau.

Imperfection du jeu : une personne attentive aura remarqué que le jeu peut tourner en rond si au moins l'un des joueurs fait osciller ses pièces sur quelques positions. Une partie sera déclarer nulle lorsque l'historique des configurations du plateau fera paraître trois occurrences d'une même configuration. Ce cas, très particulier, devra être traité en toute fin par le programmeur.

4 Une modélisation pour le « *gameplay* » au clavier

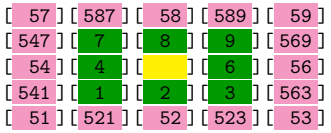
Nous allons à présent nous concentrer sur la façon dont le joueur spécifiera ses choix au clavier. Pour cela, nous ferons l'hypothèse que le jeu se fera à l'aide des touches de valeurs non-nulles du pavé numérique d'un clavier ordinaire, similaire à ceci :



Explicitons le déroulement du jeu, que l'on appelle le « *gameplay* ». Lorsque le tour d'un joueur arrive, l'utilisateur doit pouvoir faire les interactions suivantes (et selon les circonstances).

D'abord, spécifier une case : pour indiquer la j-ième case de la i-ième ligne , le joueur n'aura qu'à taper son choix avec l'entier « coordonnée » ij puis valider (donc l'entier $ij = i \times 10 + j$). Le plateau du jeu étant régulier, on remarque que i ou j sont dans l'ensemble $\{1, 2, 3, 4, 5, 6, 7\}$. Avec cette notation, la case 11 désignera celle en haut à gauche, et la case 77 celle en bas à droite dans les réprésentations graphiques précédentes.

Puis, spécifier un mouvement : pour indiquer un mouvement, il faut déterminer une case se trouvant dans l'un des deux voisinages de la case choisie. Plutôt que de saisir de nouveau une case avec un entier « coordonnée » ij, nous préférons considérer un autre systeme. Pour cela, reprenons notre figure illustrant les deux types de voisinnage et ajoutons pour chaque case dans le voisinage de la case courante (ici en jaune), un code clavier qui sera tapé pour désigner la case ciblée.



L'intuition sous-jacente est que si notre pièce choisie est représentée par la touche 5 du pavé numérique, alors toutes les touches fléchées du pavé numérique permettent de décrire naturellement tous les mouvements vers une case du voisinage. Ainsi, pour atteinre une case du voisinage

immédiat, il suffira de taper sur l'une des huit touches $\{1, 2, 3, 4, 6, 7, 8, 9\}$.

distant, il suffira de taper sur les entiers

— $\{51, 52, 53, 54, 56, 57, 58, 59\}$ pour d'étirer les mouvements précédemment décrits ;

— $\{589, 569, 563, 523, 521, 541, 547, 587\}$ pour compléter les mouvements étirés allant dans l'une des directions cardinales (les codes étant ici énoncés en balayant ce cadran dans le sens horaire).

Ou, éventuellement, abandonner un coup : l'abandon d'un coup consistera pour le joueur à taper un mouvement impossible. Votre programme devra ensuite redemander au joueur courant de vouloir saisir une position (puis un mouvement).

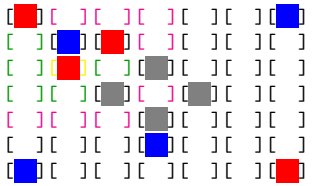
5 Rendu visuel pour le « *gameplay* »

Invite du jeu et messages : pour indiquer clairement qui doit jouer, votre jeu indiquera des messages claires au joueurs, tels que ici pour le joueur rouge :

```
[rouge] - Saisir une position à jouer :
[rouge] - Position saisie incorrecte.
[rouge] - Saisir un mouvement à réaliser :
[rouge] - Ce mouvement est invalide.
[rouge] - infecte l'adversaire !
[rouge] - passe son tour.
```

Vous pourrez définir d'autres messages, notamment pour mieux guider le joueur lorsqu'il aura commis une erreur de saisie (*e.g* lorsque la case est inexistante, ou déjà occupée...).

Visualisation des choix de voisinage : afin d'aider le joueur dans son choix pour les mouvements, et sans alourdir la présentation, le jeu devra proposer un affichage adapté comme ci-dessous dès que le joueur aura sélectionné l'une de ses pièces pour jouer un coup :

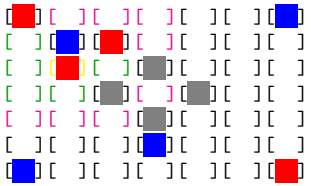


Ici, pour jouer son coup, le joueur rouge a sélectionné la pièce se trouvant en troisième ligne et deuxième colonne. La case initiale est alors colorée en jaune, celles du voisinage immédiat en vert, et celles du voisinage distant en magenta.

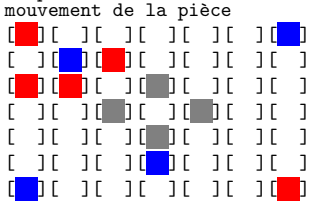
Animation d'une infection : lorsqu'un coup provoque une infection des pièces du voisinage immédiat de la pièce déplacée, une animation pourra être faite [1] (en utilisant la méthode `sleep()` de la classe `Thread`, au sein d'un petit bloc `try/catch` approprié).

1. *a minima* un simple message sera affiché à l'écran.

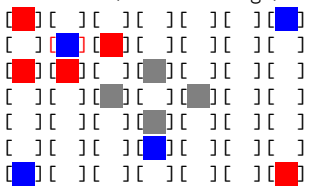
Étape 1 :
saisie d'un mouvement



Étape 2 :
mouvement de la pièce



Étape 3 (+2s):
infection (cf. case rouge)



Étape 4 (+2s) :
infection completée