Aix-Marseille Université

M3105 - Conception et programmation objet avancées

Jeu de Pente

Auteurs :
Lucien Aubert
Auguste Taillade

Enseignant :
Sébastion Thon

Table des matières

1 Introduction

Ce projet a été développé en Java dans le cadre du module M3105 de notre DUT Informatique.

1.1 Présentation du projet

Il s'est agit de développer un programme permettant de jouer une partie de jeu de Pente[1] à deux joueurs sur une même machine, jouant à tour de rôle à la souris.

Le plateau du jeu de Pente, le *goban*, ainsi que la manière d'y poser ses pions sont les mêmes qu'au jeu de Go[2] (ou *wéiqí* en chinois). Le Pente est en fait une version très simplifiée du Go dans laquelle il suffit d'aligner cinq pions de sa couleur ou de manger dix pions adverses pour gagner la partie, contrairement au Go qui laisse aux joueurs le soin de se mettre d'accord pour mettre fin à la partie.

2 Analyse

Le programme se compose deux classes qui héritent de widgets Swing.

- `Board`, qui hérite de `JPanel`
- `Cell`, qui hérite de `JButton`

La `Board` contient 361 `Cell` qui représentent les pions.

C'est cette `Board` qui applique les règles du jeu (victoire, capture de pions).

À chacun des pions sont liés des événements sous la forme de surcharges de méthodes, afin de gérer le clique et le déplacement du curseur sur ces pions. Ces événements appellent les méthodes de `Board` afin de valider le coup et agissent en conséquence : le pion est joué, le pion reste grisé, le pion n'est pas affiché ...

3 Réalisation

3.1 Choix techniques

Nous avons choisi d'utiliser Swing[3], bibliothèque graphique présente dans les JFC[4] car elle est plus performante que AWT [5]. De plus, elle est équipée d'une interface ChangeListener[6] qui permet de gérer le survol d'un élément graphique en toute simplicité, comme n'importe quel autre événement utilisateur.

Les classes du programme héritent de celles de Swing, ce qui permet leur personnalisation. Ainsi le comportement et l'apparence des widgets utilisés ont été modifiés, comme les pions, par exemple, qui sont dessinés procéduralement et non stockés sous forme d'image. Pour ce faire, la méthode `Component::paintComponent()` a été surchargée.

3.2 Vérification des alignements de pions

L'algorithme de recherche d'alignement de pions nous a demandé un effort tout particulier.

```
1  public int[] checkAlignement(Cell source, int player) {
2    /*
3    On renvoie un tableau de quatre valeurs qui correspondent  quatre
         directions (Nord>Sud, Nord-Est>Sud-Ouest, Est>Ouest, Sud-Est>Nord-
         Ouest).
4
5    Les valeurs de ce tableau indiquent pour
6    -1    Rien
7    0   Alignement de cinq pions
8    1   Alignement de capture
9    2   Alignement de capture (sens inverse)
10   */
11   int[] res = { -1, -1, -1, -1 };
12
13   int v = 0, h = 1;
14
15   /*
16   On boucle sur les quatres directions choisies (cf. commentaire au bas de
         ce code)
17   */
```

```
18    for(int i = 0; i < 4; i++) {
19      if(!(v == 1 && h == 1)) {
20        int n = 5;
21        int x, y;
22        int winCount = 1;
23        int eatCount = 0;
24        boolean sourcePassed = false;
25
26        /*
27        On va du point le plus eloigne en aval (n=5)
28        jusqu'au point le plus eloigne en amont (n=-5)
29        */
30        while(n > -4) {
31          n--;
32
33          /*
34          On calcule notre position en fonction de nos indicateurs de
                direction
35          */
36          x = source.iX + (h - 1) * n;
37          y = source.iY + (v - 1) * n;
38
39          // On ne s'attarde pas hors du plateau
40          if(x > 18 || 0 > x || y > 18 || 0 > y)
41            continue;
42
43          // On selectionne le pion a verifier
44          Cell cell = this.getCellAt(x, y);
45
46          /*
47          On teste si le pion appartient au joueur qui vient de jouer
48          */
49          if(cell == source || cell.getColor() == player) { // Pawn is ours
50            /*
51            On teste si le pion correspond au premier
52            ou au dernier pion d'un alignement de capture
53            */
54            if(
55              sourcePassed && eatCount == 3 ||
56              cell != source && !sourcePassed && eatCount == 0 ||
57              cell == source && (eatCount == 3 || eatCount == 0)
58            ) {
59              eatCount++;
60            } else {
61              eatCount = 0;
```

```
62          }
63          winCount++;
64        } else {
65          /*
66          On teste si le pion correspond au second ou au troisieme pion d'
                un
67          alignement de capture
68          */
69          if(cell.isPlayed() && (eatCount == 1 || eatCount == 2))
70            eatCount++;
71          else
72            eatCount = 0;
73          winCount = 0;
74        }
75
76        /*
77        On remplit le tableau de retour avec le resultat de la recherche
              pour la direction i
78        */
79        if(eatCount >= 4) {
80          // On renvoie le sens de l'alignement par rapport au pion joue
81          res[i] = (sourcePassed ? 1 : 2);
82        }
83
84        if(winCount >= 5) { // Bravo ! Victoire !
85          res[i] = 0;
86        }
87
88        if(cell == source)
89          sourcePassed = true;
90      }
91    }
92
93    /*
94    On change de direction :
95    initialement (i=0) la direction correspond  un trajet Nord > Sud
96    Lorsque l'on change de direction, dans le sens des aiguilles d'une
          montre, on se retrouve en Nord-Est > Sud-Ouest, puis en Est > Ouest
          , et enfin en Sud-Est > Nord-Ouest.
97    Nul besoin de verifier l'hemicycle oppose car nous prenons deja soin
          de verifier
98    4 pions de chaque cote du pion joue.
99    */
100   if(i % 3 == 0)
101     h = (h + 1) % 3;
```

```
102       else
103         v = (v + 1) % 3;
104     }
105
106     return res;
107   }
```

4 Utilisation

Le programme étant distribué sous forme d'une archive JAR exécutable aucune installation n'est nécessaire. Sous un système GNU/Linux vous pouvez lancer la commande `java -jar Pente.jar`

5 Conclusion

5.1 Optimisations possibles

La gestion des parties par le réseau n'a pas pu être fini à cause de difficultés dans la manipulation des sockets. Elle peut être viabilisée.

On peut trouver une implémentation plus propre pour le survol des pions.

5.2 Extensions possibles

Améliorer la création et l'accès aux/de parties en ligne (un pool de parties en cours/en attente d'adversaire, un mode observateur, un classement en ligne)

Références

[1] Wikipedia. Jeu de pente.
https://fr.wikipedia.org/wiki/Pente_(jeu).

[2] Wikipedia. Jeu de go.
https://fr.wikipedia.org/wiki/Go_(jeu).

[3] Wikipedia. Swing.
https://fr.wikipedia.org/wiki/Swing_(Java).

[4] JFC. Java foundation classes.
https://en.wikipedia.org/wiki/Java_Foundation_Classes.

[5] Wikipedia. Abstract window toolkit.
https://fr.wikipedia.org/wiki/Abstract_Window_Toolkit.

[6] Oracle. Interface changelistener.
https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/event/ChangeListener.html.