Licence Informatique 1^e année
Algorithmique et Programmation

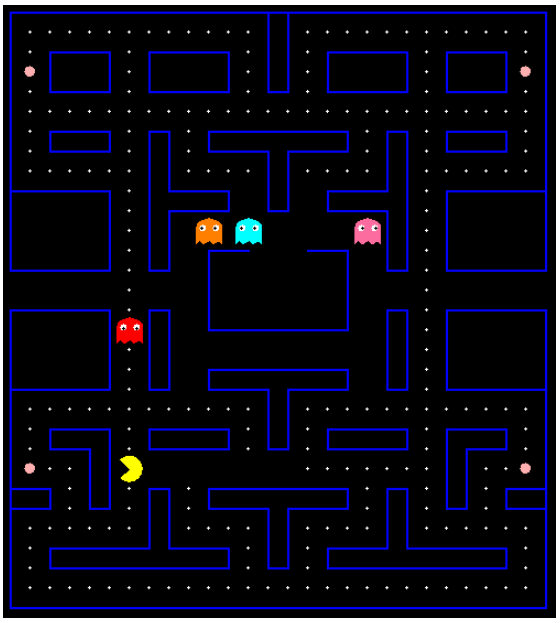
Projet 2013-2014

1 Organisation

Le projet est à réaliser en binôme. En fin de semestre, les binômes présenteront leur travail aux encadrants de TP. Lors des présentations, chaque binôme rendra un rapport qui présentera le programme réalisé (structures de données utilisées, principe des algorithmes implémentés, ...), les résultats obtenus, les problèmes rencontrés, les solutions apportées et tout élément permettant d'évaluer le travail réalisé. Chaque binôme devra également, au moment des soutenances, envoyer le code des programmes écrits aux responsables de TP (*lei.wu@u-picardie.fr*, *marisela.hernandez@u-picardie.fr*, *gil.utard@u-picardie.fr* et *frederic.furst@u-picardie.fr*).

2 Sujet

Le projet consiste à réaliser un jeu de Pacman, un classique du jeu vidéo[1].



Dans ce jeu, le Pacman, représenté par un camenbert jaune, est controlé par le joueur et se promène dans un labyrinthe où il doit manger les fruits qui se

1. *fr.wikipedia.org/wiki/Pac-Man*

trouvent le long des couloirs. Il doit éviter de se faire manger par les fantômes qui se promènent aussi dans le labyrinthe et sont contrôlés par le programme. Certains fruits bonus, lorsqu'ils sont mangés par le Pacman, donnent, pendant quelques secondes, le droit au Pacman de manger les fantômes, qui deviennent alors bleus. Les fruits et les bonus donnent des points au Pacman lorsqu'il les mange. Un niveau est terminé lorsque tous les fruits et bonus ont été mangés. Le Pacman possède plusieurs vies et à chaque fois qu'un fantôme le rejoint, il se fait manger (donc perd une vie), mais, s'il lui reste au moins une vie, le jeu redémarre avec le Pacman et les fantômes replacés à leurs positions initiales pour le niveau.

La partie interface graphique n'est pas à votre charge. Une classe Java *InterfacePacman.java* est fournie sur la page du module. Son mode d'emploi est donné plus bas. Vous devez coder tout le reste.

Remarque importante : des programmes de Pacman écrits en Java peuvent surement être trouvés sur le Web, mais ils seront écrits en Java objet. Aucun projet écrit dans le paradigme objet ne sera pris en compte lors des soutenances.

2.1 Fonctionnalités à réaliser obligatoirement

- Votre programme doit permettre de jouer à Pacman sur au moins un niveau. Le labyrinthe du niveau 1 est fourni (voir la page web du module). Vous pouvez choisir de stocker le ou les labyrinthe(s) "en dur" dans votre programme (c'est-à-dire en initialisant une structure de données dans le code) ou dans des fichiers à part qui sont chargés au besoin.
- Votre programme doit permettre au joueur de contrôler le déplacement du Pacman dans le labyrinthe, le contrôle des fantômes étant assuré par le programme. Les fantômes peuvent cependant n'avoir qu'un déplacement aléatoire.
- Votre programme doit afficher le nombre de vies et les points accumulés par le Pacman. Quand le Pacman perd sa dernière vie, il faut évidemment que le programme affiche un message *GAME OVER*, sinon ce n'est pas un vrai Pacman.

2.2 Fonctionnalités à réaliser optionnellement

Ces extensions du programme de base apportent des points en plus au projet **à condition que les fonctionnalités obligatoires aient été réalisées**.

- Vous pouvez gérer les données du jeu à l'aide de fichiers. La topographie des labyrinthes peut ainsi être stockée dans des fichiers, les noms et scores des joueurs également.
- Votre pouvez donner aux fantômes des comportements différents et plus ou moins agressifs envers le Pacman. Par exemple, un fantôme peut systématiquement chercher à manger le Pacman en empruntant le plus court chemin pour le rejoindre (voir plus bas quelques éléments pour le calcul de ce plus court chemin). Un autre fantôme peut par exemple se promener en privilégiant la ligne droite (il ne change de direction que quand il ne peut faire autrement). Etc.

- Vous pouvez ajouter d'autres niveaux que le niveau 1, en créant de nouveaux labyrinthes et/ou en modifiant la vitesse du jeu, les points obtenus en mangeant les fruits, etc.
- Vous pouvez ajouter de la musique[2].

2.3 Démarche générale pour programmer le Pacman

La première chose à faire est de comprendre le jeu, en jouant. Il faut ensuite bien identifier les données à manipuler :

- Quelles données sont nécessaires pour représenter le labyrinthe?
- Quelles données sont nécessaires pour décrire le Pacman?
- Quelles données sont nécessaires pour décrire les fantômes?
- Quelles données sont nécessaires pour décrire l'état du jeu (score, etc)?

Une fois les données à représenter identifiées, vous devez choisir des structures de données à utiliser. Par exemple, il parait incontournable d'utiliser un tableau à deux dimensions pour représenter le labyrinthe. Mais on peut choisir de coder le contenu de chaque case avec des nombres, ou avec des enregistrements ou autre chose. Autre exemple, le Pacman étant décrit au moins par sa position, sa direction, son nombre de vies, son aspect (bouche ouverte ou fermée) il parait pratique de le représenter par un enregistrement.

Une fois les structures de données choisies, vous pouvez déjà écrire le code Java correspondant à la déclaration de vos structure et à leur initialisation. À ce stade, vous pouvez déjà utiliser l'interface graphique pour afficher l'état initial du jeu.

L'étape suivante est d'écrire l'algorithme qui fait tourner le jeu. Le mieux est d'y aller étape par étape. Par exemple, commencez par écrire un algorithme qui permettra de faire bouger le Pacman, puis qui permettra en plus à l'utilisateur de modifier sa direction. Ajoutez ensuite la gestion des fantômes.

L'idée générale de l'algorithme du jeu est la suivante :

```
tant que la partie n'est pas perdue faire
   récupérer la touche tapée par l'utilisateur
   faire bouger le Pacman
   gérer la collision éventuelle du Pacman avec les fantômes
   faire bouger les fantômes
   gérer la collision éventuelle des fantômes avec le Pacman
fintantque
```

Pour gérer la vitesse du jeu, on peut utiliser l'instruction `try{Thread.sleep(n);} catch(InterruptedException e){}` qui met le programme en attente `n` millisecondes.

2. Voir le tutoriel *http://docs.oracle.com/javase/tutorial/sound/*

2.4 Calcul du plus court chemin

Le problème du plus court chemin dans un graphe, un réseau ou un labyrinthe a été l'objet de nombreux travaux de recherche en algorithmique. Le mieux à faire est donc d'utiliser des algorithmes existants et éprouvés. Un algorithme assez simple et efficace est l'algorithme de Dijkstra. En voici une version adaptée pour calculer, dans le labyrinthe du Pacman, le plus court chemin entre deux cases. Pour bien comprendre cet algorithme, vous pouvez demander des explications aux enseignants et/ou consulter des livres ou des sites Web.

L'idée de l'algorithme est de calculer par itérations successives la distance minimale entre la case de départ (qu'on va appeler `A`) et la case d'arrivée (qu'on va appeler `B`). Pour cela, on calcule pour chaque case franchissable la distance qui la sépare de `A`. On commence par les cases qui entourent `A` (qui sont à une distance de 1), puis celles qui entourent les précédentes (qui sont à une distance de 2), et ainsi de suite jusqu'à ce qu'on arrive à `B`. Comme il peut exister plusieurs chemins conduisant à une case, si on tombe sur une case pour laquelle on a déjà calculé une distance, il faut comparer la distance déjà calculée, avec la distance à laquelle on aboutit en suivant le chemin qu'on est en train de calculer. Si ce nouveau chemin est plus court, la nouvelle distance devient la distance de la case. Il faut donc mémoriser pour chaque case le chemin de longueur minimum permettant d'atteindre cette case. En pratique, il suffit de mémoriser pour chaque case la case qui la précède sur ce chemin.

À chaque case est donc attachée une marque (qui permet de savoir si on a déjà franchi cette case ou pas), une distance minimale pour l'atteindre et la case qui la précède sur le chemin le plus court. On appelle DM(x) la distance minimale de la case `x` et `prec(x)` la case qui précède la case `x`.

```
initialiser toutes les cases franchissables comme non marquées, avec
           une distance minimale de +∞ et aucune case précédente
initialiser la case de départ A avec une distance minimale de 0
tant que la case B n'a pas été marquée faire
   soit C la case non marquée avec la plus petite distance minimale
   marquer C
   pour chaque case D adjacente à C, franchissable et non marquée faire
      si DM(D) > DM(C) + 1 alors
         DM(D) <- DM(C) +1
      finsi
   fin pour
fin tant que
```

Une fois cet algorithme exécuté, pour reconstituer le plus court chemin, il suffit de remonter à partir de la case `B` vers la case `A` à l'aide des liens entre cases précédentes. On peut même simplement retrouver la case qui suit immédiatement la case `A` dans ce plus court chemin : c'est la case par où le fantôme doit passer pour atteindre le Pacman le plus vite possible.

3 L'interface d'affichage

Du code est fourni pour l'affichage du jeu. Il est obligatoire de l'utiliser. Pour l'utiliser, placez simplement le fichier *InterfacePacman.java* dans le répertoire où se trouve votre programme, vous pourrez alors utiliser dans votre programme les instructions données plus bas. Il est inutile de lire le contenu de cette classe, et encore moins de le comprendre, pour l'utiliser. Les explications qui suivent suffisent pour l'utiliser.

3.1 Les directions

Les directions sont représentées dans l'interface par des constantes entières. Les constantes de direction sont *WEST, EAST, NORTH, SOUTH* et *CENTER* (utilisée pour indiquer une immobilité). Elles sont définies dans la classe *javax.swing.SwingConstants*. Vous pouvez les utiliser dans votre programme, en particulier comme paramètre de certaines fonctions de l'interface. Pour les utiliser dans votre programme, deux possibilités :

- ajouter *import javax.swing.SwingConstants* en tête de votre programme. Les constantes de direction peuvent alors être utilisées en les préfixant (par exemple *SwingConstants.NORTH*)
- ajouter *import static javax.swing.SwingConstants* en tête de votre programme. Les constantes de direction peuvent alors être utilisées sans préfixe (par exemple *NORTH*).

3.2 Les couleurs

Les couleurs sont représentées dans l'interface par les valeurs du type *Color*. Des constantes sont définies dans la classe *java.awt.Color*: *BLUE, RED, GREEN, BLACK, WHITE,* etc. Pour utiliser ce type dans votre programme, ajoutez *import java.awt.Color* en tête de votre programme (ou *import static java.awt.Color*). Pour plus d'informations, voir la documentation Java[3].

3.3 Les instructions disponibles pour utiliser l'interface

Instruction pour créer une interface : `InterfacePacman toto = new InterfacePacman(l, h);` où `l` et `h` sont respectivement la largeur et la hauteur du labyrinthe (en nombre de cases).

Instruction pour récupérer la touche appuyée par le joueur : `int touche = toto.toucheAppuyee();` où `toto` est la variable contenant l'interface créée. L'entier renvoyé sera *EAST* si le joueur a tapé sur la flèche droite, *WEST* s'il a tapé la flèche gauche, *NORTH* s'il a tapé la flèche haut, *SOUTH* s'il a tapé la flèche bas, *CENTER* s'il a tapé la touche pause et -1 s'il a tapé une autre touche.

Instruction pour effacer une case (c'est-à-dire la remplir avec la couleur de fond) : `toto.effaceCase(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées de la case.

3. *docs.oracle.com/javase/7/docs/api/*

Instruction pour dessiner un obstacle : `toto.dessinerObstacle(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées de l'obstacle.

Instruction pour dessiner un fruit : `toto.dessinerFruit(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du fruit.

Instruction pour dessiner un bonus : `toto.dessinerBonus(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du bonus.

Instruction pour dessiner un Pacman bouche fermée : `toto.dessinerPacmanFerme(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du Pacman.

Instruction pour dessiner un Pacman bouche ouvert : `toto.dessinerPacmanOuvert(x,y,d)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du Pacman et `d` est la direction du Pacman (EAST, WEST, SOUTH ou NORTH).

Instruction pour dessiner un fantôme non mangeable : `toto.dessinerFantome(x,y,c,d)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du fantome, `c` la couleur du fantôme (valeur du type `Color` et `d` est la direction du fantôme (EAST, WEST, SOUTH ou NORTH).

Instruction pour dessiner un fantôme mangeable : `toto.dessinerFantomeMangeable(x,y)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du fantôme.

Instruction pour afficher un message dans une boite de dialogue : `toto.afficheMessage(m)` où `toto` est la variable contenant l'interface créée et `m` la chaine de caractères à afficher.

Instruction pour afficher un texte dans la zone de texte en bas de l'interface : `toto.afficheTexte(m)` où `toto` est la variable contenant l'interface créée et `m` la chaine de caractères à afficher.