Licence Informatique 1^e année
Algorithmique et Programmation

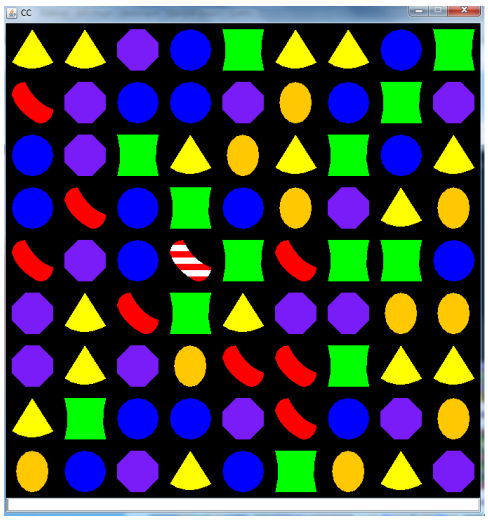
Projet 2014-2015

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* et *frederic.furst@u-picardie.fr*).

2 Sujet

Le projet consiste à réaliser un jeu de type Beejeweled [1] ou Candy-Crush [2].



Le principe du jeu est d'échanger deux bonbons (ou gemmes) de place, pour obtenir des alignements de 3 (ou plus) bonbons de même type. Un échange n'est possible que lorsqu'il crée un alignement d'au moins 3 bonbons de même type. Quand un alignement est réalisé, les bonbons impliqués dans l'alignement disparaissent et les bonbons situés au dessus descendent d'un cran. Quand il n'y a

1. *http ://bejeweled.popcap.com/html5/*
2. *http ://www.candy-crush-saga.fr/*

plus d'échange possible, le jeu est terminé. Lorsqu'un alignement de 4 bonbons de même type est réalisé, un des bonbons qui disparait est remplacé par un bonbon bonus de même type. Lorsqu'un de ces bonbons bonus se retrouve dans un alignement, toute la ligne disparait s'il s'agit d'un alignement horizontal ou toute la colonne s'il s'agit d'un alignement vertical.

Il est également possible de faire apparaitre des bonbons super bonus lorsque 5 bonbons sont alignés, et ces super bonus permettent de réaliser d'autres actions (voir la notice du jeu en ligne).

La partie interface graphique n'est pas à votre charge. Une classe Java *InterfaceCC.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 CandyCrush 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 CandyCrush, avec au moins les bonbons bonus. Les alignements doivent être correctement détectés (qu'ils soient créés par le joueur ou qu'ils résultent de l'apparition de nouveaux bonbons en haut de l'écran). Le score du joueur doit évoluer en fonction du nombre et du type de bonbons éliminés. Le score du joueur peut être affiché en fin de partie, ou durant la partie sur l'interface.

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 liées au jeu à l'aide de fichiers. Ces données peuvent concerner les parties (sauvegarde d'une partie en cours, chargement d'une partie sauvegardée) et/ou les joueurs (sauvegarde des meilleurs scores de chaque joueur, de l'historique des parties de chaque joueur, ...).
- Votre pouvez gérer les bonbons super bonus, créer d'autres types de comportements que vous inventez. Par exemple, si le joueur arrive à créer un carré de 4 bonbons, cela fait apparaitre un bonus qui, une fois activé, fera disparaitre tous les bonbons dans un rayon de 3 cases. Si nécessaire, des apparences graphiques supplémentaires pour les bonbons peuvent être ajoutées sur demande dans l'interface.
- Vous pouvez créer des niveaux de jeu. Passer d'un niveau à l'autre peut se faire en atteignant un certain seuil de points, ou après un certain temps de jeu. Chaque niveau peut avoir des spécificités (effets différents pour les bonus, etc).
- Vous pouvez ajouter de la musique[3].

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

2.3 Démarche générale pour programmer le jeu

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 la grille du jeu et les bonbons ?
- Quelles données sont nécessaires pour décrire le joueur, son 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 la grille. Mais on peut choisir de coder le contenu de chaque case avec des nombres, ou avec des enregistrements ou autre chose.

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 permet à l'utilisateur de sélectionner deux cases et qui échange leur contenu. Ajoutez ensuite la détection des alignements de bonbons, puis leur disparition.

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.

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 *InterfaceCC.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 instructions disponibles pour utiliser l'interface

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

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.

Instruction pour dessiner un bonbon carré vert : `toto.dessinerCarre(x,y,n)` où `toto` est la variable contenant l'interface créée et `x` et `y` sont les abscisses et ordonnées du bonbon et `n` est le niveau du bonbon (si `n` vaut 1 le bonbon est

dessiné plein, si `n` vaut 2 il est dessiné rayé de blanc, si `n` vaut 3 il est dessiné avec des cercles blancs).

Instructions pour dessiner les autres bonbons : fonctions avec les mêmes paramètres nommées `dessinerOvale` (en orange), `dessinerLarme` (en jaune), `dessinerRond` (en bleu), `dessinerHexagone` (en violet), `dessinerSaucisse` (en rouge).

Instruction pour savoir quelle case l'utilisateur a cliquée : `toto.clicCase()` renvoie un `Point` avec pour champs `x` et `y` les coordonnées de la case cliquée.

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.