

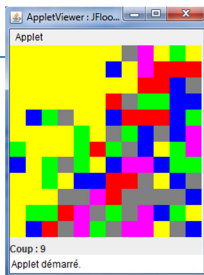
JFloodIt

CREATION : 2011/04/04

MISE A JOUR : 2011/04/04

Notions : JApplet, déploiement

Nous vous proposons de coder une implémentation java de Flood It, le jeu publié par LabPixies sur iPhone et Android.



Partie 1 : Principe de jeu

Le principe est très simple : Vous disposez d'une zone de jeu carrée de 12 cases par 12 cases. Chacune des cases est remplie d'une couleur sur 6 possibles. Vous devez vous arranger pour que toutes les cases soient de la même couleur. Pour cela, vous choisissez une couleur pour la case en haut à gauche et cela peint toutes les cases adjacentes de cette couleur : le raz de marée. On dira qu'une case est adjacente si elle est de la couleur de la première case avant le changement et si elle touche une case de cette couleur soit par le haut/bas, soit par la gauche ou la droite (on ne considère pas les cases en diagonales).

La zone doit être couverte en 22 coups au maximum. Voici un petit exemple. On remplace les couleurs par un chiffre entre 0 et 5. La première case est de la couleur 0. A l'étape 1, on choisit la couleur 1 (On aurait pu choisir 2 : une seule case aurait été couverte en plus mais choisir 3 ou 4 n'aurait pas eu d'intérêt). A l'étape 2, on choisit la couleur 3 ... Vous avez compris ?

0	1	3	1
2	1	2	2
3	1	1	3
0	0	4	4

Départ

1	1	3	1
2	1	2	2
3	1	1	3
0	0	4	4

Étape 1

3	3	3	1
2	3	2	2
3	3	3	3
0	0	4	4

Étape 2

Partie 2 : implémentation

1. Déclarer une classe JFloodIt qui dérive de JApplet. La classe JApplet exécute une fois la méthode init() et peut exécuter plusieurs fois la méthode start() (si l'applet est relancée). Ces méthodes doivent être les plus rapides possibles.
2. Déclarer un attribut zone de type int [][] dans la classe JFloodIt
3. Initialiser zone dans la méthode init() à un carré de dimension 12.
4. Mettre des valeurs aléatoires entières comprises entre 0 et 5 pour chaque élément de la zone dans la méthode start(). Afficher sur la console que c'est bien fait.
5. Déclarer une classe Zone qui étend JPanel. Cette classe sera la représentation graphique la zone précédente.
6. La classe Zone utilise un attribut zone, de même type que celui déclaré au 2. Le lien sera fait soit par un setter dans init(), soit à la construction.
7. Redéfinir la méthode paintComponent() de Zone pour afficher un petit rectangle.

8. Créer une méthode creerGUI() dans la classe JFloodIt. Cette méthode ajoute une instance de Zone au getContentPane().
9. Pour des questions de timing (et de confinement), l'appel à creerGUI() doit être « protégé ». Vous vous souvenez comment on a créé l'interface graphique dans le tutoriel Swing ? Il faut faire la même chose en utilisant la méthode SwingUtilities.invokeLaterAndWait() à partir d'init(). Cela permet de s'assurer que l'interface graphique est bien créée dans le thread des événements de Swing et que l'interface est opérationnelle à la fin de la méthode init().
10. Créer un tableau de Color statique dans la classe Zone avec six couleurs de votre choix.
11. Modifier Zone.paintComponent() pour que la zone de jeu soit correctement affichée.
12. On y presque ! On doit maintenant interagir avec l'utilisateur. Je vous propose d'implémenter un MouseListener au niveau de JFloodIt.
13. Redéfinir la méthode concernant le clic. Il faut calculer les coordonnées (i,j) de la grille en fonction des coordonnées souris (getX(), getY()). Il faut ensuite appeler la méthode remplir (zone, 0, 0, zone[0][0], zone[i][j]) et lancer un repaint(). C'est déjà pas mal, non ?

Voici le code de la méthode remplir() :

```
static boolean remplir(int[][] grille, int i, int j,
                      int ancien, int nouv) {
    if ((i>=TAILLE) || (j>=TAILLE) || (i<0) || (j<0))
        return false;
    if ((i==0) && (j==0) && nouv == ancien)
        return false;

    if (grille[i][j] == ancien) {
        grille[i][j] = nouv;
        remplir(grille, i, j+1, ancien, nouv);
        remplir(grille, i+1, j, ancien, nouv);
        remplir(grille, i-1, j, ancien, nouv);
        remplir(grille, i, j-1, ancien, nouv);
    }
    return true;
}
```

14. On ajoute une barre des tâches, un composant JLabel attribut de JFloodApp. Le composant affiche une petite aide au départ, le nom de coups en cours de route. Un composant de type JLabel n'est pas opaque par défaut, alors cela peut être une bonne idée qu'il le soit.

Partie 3 : déploiement

Pour tous les tests que vous avez faits, Eclipse a lancé appletviewer. Nous allons maintenant nous intéresser au déploiement de cette applet.

15. Avec un clic droit sur le répertoire du projet, choisir d'exporter les fichiers sources sous forme d'un fichier jar (celui-ci n'a pas besoin d'être exécutable). Il faut choisir le nom et le répertoire de destination. Pour la suite, le fichier est supposé s'appeler JFloodIt.jar



16. Placer dans le même répertoire que le fichier jar, un fichier html avec la balise suivante

```
<applet code="JFloodIt"
        codebase="." archive="JFloodIt.jar"
        width="300" height="300">
</applet>
```

17. Le résultat peut apparaître maintenant dans un navigateur web !

Ce n'est pas le seul moyen de déployer une applet mais c'en est un ! Oracle préconise que l'on utilise un script javascript `deploy.js` pour s'affranchir des différences entre les navigateurs. On peut modifier l'applet pour qu'elle soit également lançable comme une application autonome. Pour déployer l'application, il faudra alors faire un jar exécutable et un simple clic permettra de lancer le petit jeu.

Partie 4 : améliorations

Que manque-t-il encore à cette application ?

- La gestion du nombre de coups (dans le jeu original, il ne faut pas dépasser 22 coups)
- Un petit menu ou un bouton pour faire une nouvelle partie
- Adapter l'application pour être aussi une application autonome. Ceci est relativement facile si l'on se rappelle qu'une `JApplet` et un `JFrame` ont en commun la méthode `getContentPane()`. Je vous propose alors de paramétrer la méthode `creerGUI()` avec un booléen et de choisir le bon `RootPaneContainer`