

Fourmiz



Introduction

Objectif

Malgré la maîtrise du feu, de la pierre ou encore de l'atome, l'Homme s'incline devant le caractère imprévisible de la nature. Néanmoins il essaye de comprendre ses finesses en créant des simulateurs toujours plus performant à l'aide du nouvel outil d'aujourd'hui: l'informatique.

Ainsi nous devons réaliser la vie d'une colonie de fourmis ou plutôt, la recherche de nourriture de cette même colonie sur un terrain accidenté.

Surface de jeu

Le terrain de jeu est un plateau rectangulaire $m \times n$. Les bords représentent la limite du plateau; il n'est pas possible de se retrouver de l'opposé du plateau (le plateau ne se comporte pas comme une sphère).

Exemple :

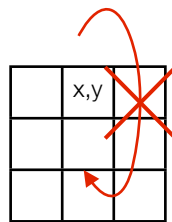
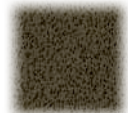


fig 1 : mouvement impossible d'une fourmis

Il existe trois cases logicielement différentes et quatre visuellement :

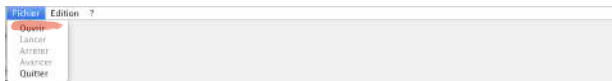
Type	Légende
	L'obstacle est infranchissable par la fourmis. Elle devra contourner cette case.
	La case steak représente la nourriture. Un libellé montre le nombre restant de nourritures sur cette case
	La case fourmilière. Les fourmis naissent à partir de cette case et ramènent également la nourriture à ces coordonnées.
	Les cases phéromones. Elles permettent de tracer le chemin de la nourriture à la fourmilière par une fourmis lorsque celle-ci à dénicher une case nourriture.



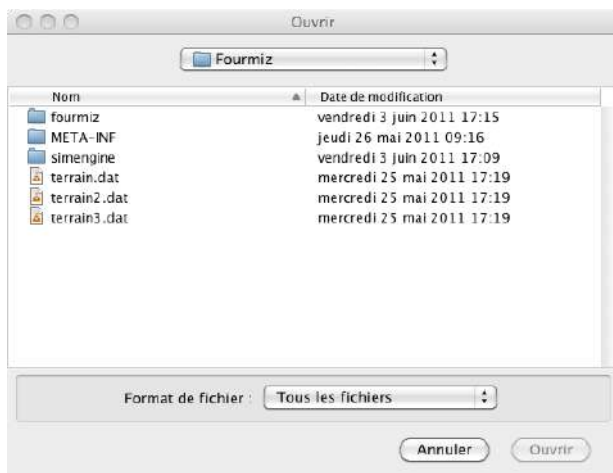
Déroulement d'une partie



Lorsque vous lancez le jeu, vous devez lancer un plateau pré-construit.



Cliquez sur fichier puis ouvrir pour sélectionner un terrain



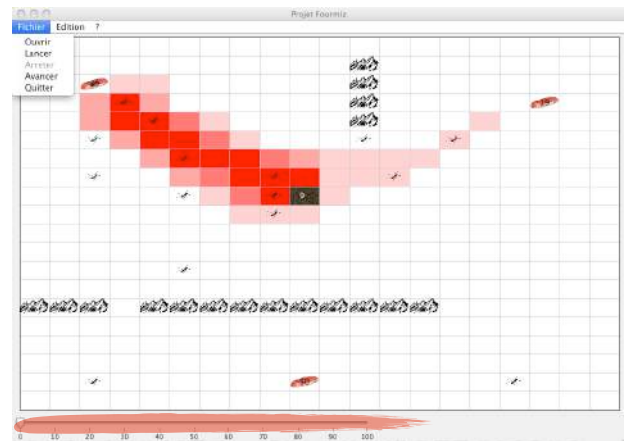
Lancer ensuite un des trois terrain disponible

terrain.dat

terrain2.dat

terrain3.dat

Le terrain chargé vous avez deux possibilités pour démarrer le jeu.

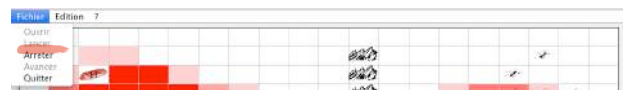


Lancer permet de démarrer la simulation en continue avec une vitesse dépendant du curseur (en rouge)

Avancer permet de démarrer la simulation step by step



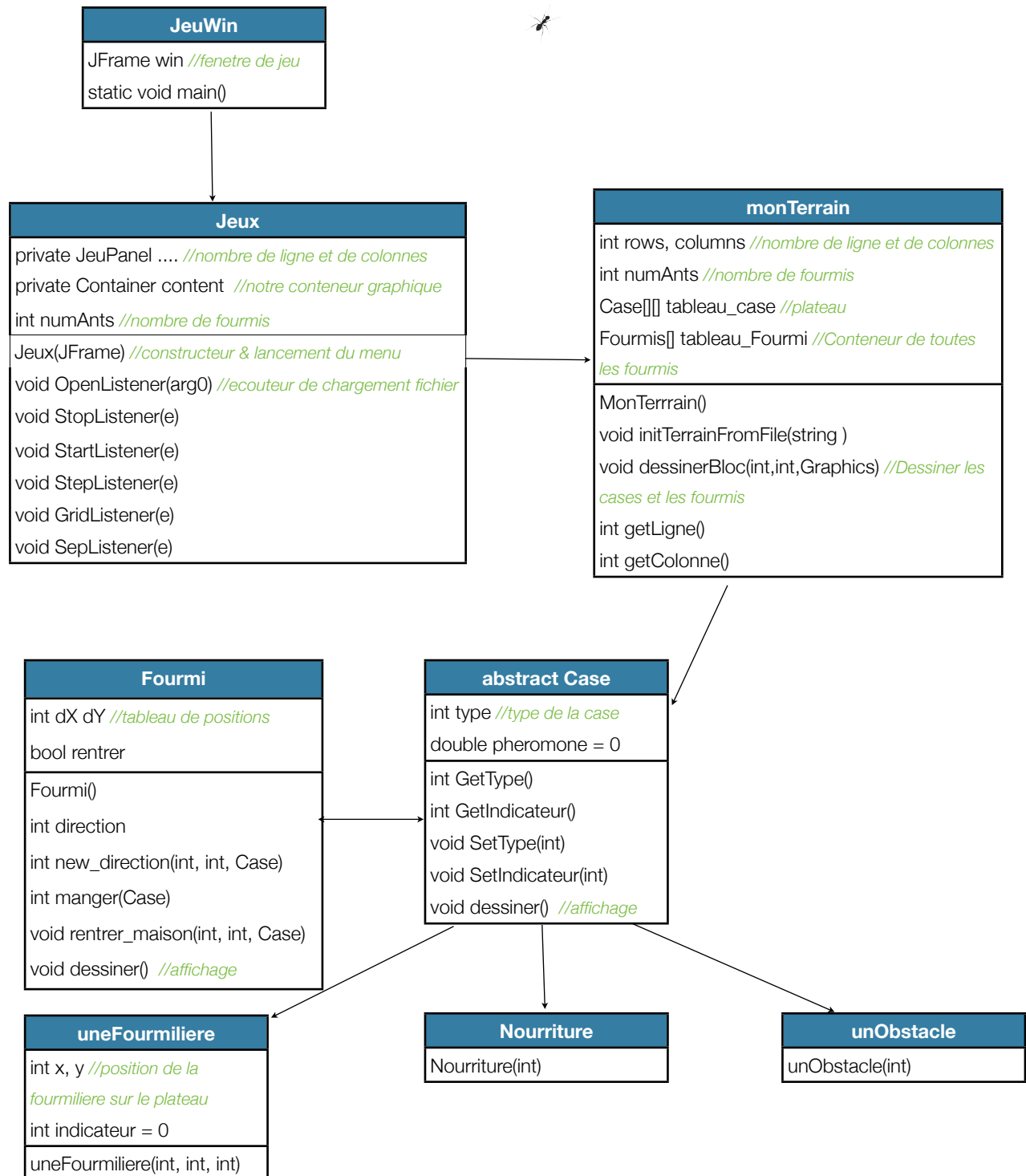
Affiche ou supprime la grille, le carrelage du plateau



Stop la simulation, met la simulation en pause.



Diagramme de classes



Aspect programmation

Nous avons choisi de gérer l'ensemble du projet (et l'option phéromone donc) avec le respect du framework imposé:

La classe JeuWin contenant main est donc très similaire à celle fournie. Le jeu se lance réellement via la création d'un objet de type Jeux, appelé par *new Jeux(win)* ou *win* est la fenêtre de jeu de type JFrame.

```
JFrame.setDefaultLookAndFeelDecorated(true); // Utiliser decorations Swing
JFrame win = new JFrame("Fourmiz"); // Créer la fenêtre avec titre
win.setTitle("Projet Fourmiz By B&T"); // B&T le meilleur binome :D
new Jeux(win); // On lance le jeu
win.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Quitter l'appui lors d'un appui sur X
win.setSize(1024,840); // Etablir les dimensions initiales
win.setLocationRelativeTo(null); // Centrer la fenêtre
win.setVisible(true); // Afficher la fenêtre
```

La classe Jeux est la classe où se construit l'interface graphique, l'ensemble des boutons (de JMenuBar) sont instanciés et leurs écouteurs définis.

Lorsque le joueur charge un fichier, le listener OpenListener fait appel à la classe MonTerrain pour afficher/instancier le plateau de jeu.

Nous avons choisi d'utiliser des JMenu plutôt que des JButton afin de simplifier l'interface.

```
/** OPEN **/
this.fichier.add(openButton);
openButton.addActionListener(new OpenListener());

/** START **/
this.fichier.add(start);
start.addActionListener(new StartListener());
start.setEnabled(false);

/** STOP ****/
this.fichier.add(stop);
stop.addActionListener(new StopListener());
stop.setEnabled(false);

/** STEP ****/
this.fichier.add(step);
step.addActionListener(new StepListener());
step.setEnabled(false);

/** EXIT **/
exit.addActionListener(new ActionListener())
```



La classe MonTerrain, comme indiqué dans le sujet, est issue de l'interface TerrainDeJeu, il y est donc défini dessinerBloc qui affiche/dessine toutes les cases et les fourmis, mais aussi et surtout, la fonction de chargement fichier du jeu qui, pour chaque case, attribue un type (obstacle, nourriture, fourmilière) dans le tableau des cases du jeu.

A noter : la classe possède aussi un tableau de fourmi, celles-ci, bien que sur des cases, ne sont pas considérées comme telles (seul obstacle, fourmilière et nourriture héritées de la classe case).

La classe Case est une interphase, aucune de ses fonction n'est défini (elle initialise juste les phéromones à zéro). Elle contient les fonctions essentielles au case : l'affichage/dessin et des accesseurs/mutateurs.

```

/** CLASSE ABSTRAITE INTERFACE ****/
public abstract class Case
{
    protected int type;
    protected double pheromone = 0;

    public abstract void dessiner (Graphics graph);
    public abstract int GetType();
    public abstract int GetIndicateur();
    public abstract void SetType(int untype);
    public abstract void SetIndicateur(int unindic);
}

```

uneFourmilere est une classe fille de Case possédant ses attributs X et Y en static : en effet il n'y a qu'une fourmilière par partie et elle se situe a une position fixe que les fourmis ont besoin de connaître, voilà pourquoi nous l'avons implémenté de cette façon.

```

public class uneFourmilere extends Case
{
    static public int PosX = 0;
    static public int PosY = 0;
    private int indicateur = 0;

    uneFourmilere(int untype, int x, int y)
    {
        PosX = x;
        PosY = y;
        //On initialise en fourmilere
        type = untype;
    }
}

```

La classe unObstacle se contente principalement d'afficher son image: cette case est fixe et n'évolue pas, les fourmis l'éviterons grâce a la valeur de son type.

Nourriture est un peu plus complexe : elle gère l'affichage de l'image et du nombre de nourriture restante sur sa case mais aussi l'affichage/diminution des phéromone (de 0,05 par step, soit 1 unité tout les 20) en se basant sur un dégradé de rouge.

```

if(pheromone >= 80)
    z = 0;
else
    if(pheromone >= 60)
        z = 100;
    else
        if(pheromone >= 40)
            z = 150;
        else
            if(pheromone >= 20)
                z = 200;
            else
                z = 255;
//Affichage en rouge dégradé
Color c = new Color(255, z, z);

```



La classe fourmi permet deux type de déplacement : celui de la recherche de nourriture, et, si la nourriture est trouvée, celui du retour a la maison.

Lorsqu'une fourmi est sur une case de nourriture, elle fait appel a sa fonction manger pour diminuer la nourriture de la case et la transporter avec elle.

Les fonction new_direction et rentrer_maison permettes de modifier la direction de la fourmi.