Wator

Jean-Christophe Routier
Licence mention Informatique
Université Lille 1

Le jeu de la vie

Un environnement est représenté par une grille torique.
Une case de la grille représente une cellule active ou pas.

Le comportement d'une cellule est le suivant :

- Une cellule inactive s'active si elle est entourée d'exactement 3 cellules actives.
- Une cellule active ne survit que si elle est entourée de 2 ou 3 cellules actives.

Toutes les cellules évoluent simultanément.

Modélisation

- l'environnement
- la dynamique de l'application
- l'affichage de l'environnement

Wator

Un modèle de simulation proie-prédateur

Un environnement est représenté par une grille torique de cases.
Dans cet environnement évoluent, selon certaines règles, des thons et des requins. On trouve au plus un poisson par case.

Fonctionnement : on tire aléatoirement une position dans l'environnement et le poisson qui s'y trouve (quand il y en a un) essaie de se déplacer puis éventuellement donne naissance à un autre poisson puis éventuellement meurt.
La dynamique de l'évolution consiste à voir évoluer dans le temps les populations respectives de requins et de thons.

Comportements

Thon (proie)

- Les thons peuvent se déplacer vers une des 8 cases voisines si elle est libre.
- Ils se reproduisent avec une certaine période : après un déplacement réussi, la descendance apparaît sur la case libérée.

Requin (prédateur)

- Les requins peuvent se déplacer vers une des 8 cases voisines si elle est libre ou si elle est occupée par un thon. Dans ce dernier cas le requin mange le thon qui "disparaît".
- La reproduction des requins suit les mêmes règles que celle des thons (avec un période a priori différente)
- Un requin meurt de faim s'il n'a pas mangé de thon avant une certaine période de temps. Dans ce cas il disparaît.

Modélisation

- l'environnement
- les thons
- les requins
- la dynamique de l'application
- l'affichage de l'environnement

Grille torique et affichage

Grille

- largeur, hauteur,
- obtenir le caractère pour une **position** donnée
- obtenir la couleur (dessin) pour une position donnée

Affichage

- afficheur texte
- afficheur "graphique"

notions **communes**, donc **réutilisées**, avec le jeu de la vie.
factorisation de fonctionnalités

Analyse

Réfléchir en terme de fonctionnalités.

```
Application :
 initialiser l'environnement
 pour i allant de 1 à n cycles
   faire évoluer l'environnement

Evolution :
 choisir un poissons p au hasard
   choisir une case destination d
   si p peut bouger vers d
     déplacer p
     si p peut donner naissance
       créer un nouveau poisson du type de p (à sa position précédente)
       remettre le temps gestation à 0
     sinon incrémenter le temps de gestation
   si p doit mourrir
     supprimer p
 afficheur.affiche(environnement)
```

Polymorphisme ?

!!! Abstraire !!!

Polymorphisme : Réutiliser

- Le jeu de la vie et wator sont modélisés par des grilles toriques et donc la représentation de l'environnement est similaire et (par exemple) les procédures d'affichage sont les mêmes.

interface `Grid`

Polymorphisme : Diversifier

- Pouvoir avoir un affichage de l'environnement en mode texte ou en mode graphique.

interface `GridDisplayer`

Polymorphisme : Généraliser

Wator L'environnement est une grille de *Poisson* qui peuvent être des thons ou des requins avec des comportements différents. L'environnement doit pouvoir manipuler les entités indifféremment.

interface `Fish`

Grid

```
package grid;
public interface Grid {
   public int getWidth();
   public int getHeight();
   public java.awt.Color getColorAtPosition(Position p);
   public char getCharAtPosition(Position p);
}// Grid
```

GridDisplayer

```
package grid;
public interface GridDisplayer {
   public void display(Grid grid);
}// GridDisplayer
```

Implémentations

```
package grid;
public class TextGridDisplayer implements GridDisplayer {
  public void display(Grid grid) {
    for (int i=0; i<grid.getWidth(); i++) {
      for (int j=0; j< grid.getHeight();j++) {
         System.out.print(grid.getCharAtPosition(new Position(i,j)));
      } // for j
      System.out.println();
    } // for i
  }
}// TextGridDisplayer
-------------------------------------------------
package grid;
public class GraphicalGridDisplayer implements GridDisplayer {
  public void display(Grid grid) {
    ...
}
}// GraphicalGridDisplayer
```

Fish

```
package wator ;
public interface Fish {
   public boolean canEat(Fish f) ;
   public Position getPosition() ;
   public FishType getFishType() ;
   public void setPosition(Position p) ;
   public void setGestationPeriod(int period) ;
   public void decreaseTimeBeforeNextBirth() ;
   public boolean canGiveBirth() ;
   public Fish createnewFish(Position position) ;
   public boolean isDead() ;
   public char getDescriptionChar() ;
   public java.awt.Color getColor() ;
}// Fish
```

```
package wator ;
public class Tuna implements Fish {
   private static final char TUNA_CHAR = 'T' ;
   ...
   public char getDescriptionChar() {
      return TUNA_CHAR ;
   }
}// Tuna
```

```
package wator ;
public class Shark implements Fish {
   private static final char SHARK_CHAR = 'S' ;
   ...
   public char getDescriptionChar() {
      return SHARK_CHAR ;
   }
}// Shark
```

Environnement

```
package wator;
import grid.*;
public class Environment implements Grid {
   private GridDisplayer gridDisplayer;
   private Fish[][] fishes;
   ...
   public void setGridDisplayer(GridDisplayer displayer) {
      this.gridDisplayer = displayer;
   }
   public void display() {
      this.gridDisplayer.display(this);
   }
   public char getCharAtPosition(Position p) {
      Fish f = this.fishes[p.getX()][p.getY()];
      try {
         return f.getDescriptionChar();
      catch(NullPointerException e) {
         return ' ';
      }
   }
```

```
    public class Environment implements Grid {
       ...
       public void evolue() {
             choisir i et j au hasard
             Fish fish = this.fishes[i][j];
             if (fish != null) {
                Position current = fish.getPosition();
                Position targetPosition = current.randomNeighbour(width, height);
                Fish targetFish = this.fishes[targetPosition.getX()][targetPosition.getY()];
                if (targetFish == null || fish.canEat(targetFish)) {
                   fish.setPosition(targetPosition);
                   this.fishes[targetPosition.getX()][targetPosition.getY()] = fish;
                   this.fishes[i][j] = null;
                   if (fish.canGiveBirth()) {
                      fishes[i][j] = fish.createNewFish(current);
                   }
                   fish.decreaseTimeBeforeNextBirth();
                   if (fish.isDead()) {
                      this.fishes[fish.getPosition().getX()][fish.getPosition().getY()] = null;
                   }
                } // else nothing happens
          }

       ...
    } // Environment

//====== et dans un "main(...)"
                Environment env = new Environment(...);
                env.evolue();
                   env.setGridDisplayer(new GraphicalGridDisplayer());
                ou env.setGridDisplayer(new TextGridDisplayer());
                env.display();
```