UNIVERSITÉ de Picardie Jules Verne

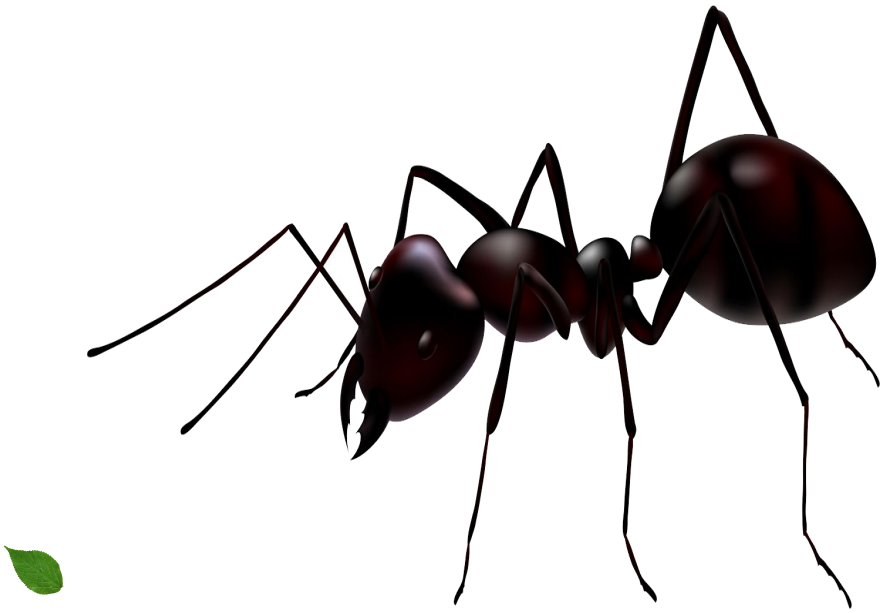
Licence Informatique – 2ème Année

Rapport de projet de programmation

Module Programmation Objet 1

Simulation d'une colonie de fourmis

Créateur: AIWANSEDO Konstandinos



Introduction

Le Rapport porte sur le projet de Programmation Objet et consiste en la simulation d'une colonie de fourmis. Le projet comporte 3 types de fourmis différents: la reine, l'ouvrière et la guerrière. La reine produit des guerrières et des ouvrières, les ouvrières se déplacent de façon aléatoire et s'occupent de la recherche des provisions, comme nourriture et matériaux et les guerrières se déplacent de façon aléatoire à la recherche de menace. Le projet comporte aussi des animaux menaçants comme des oiseaux et des araignées qui se déplacent de façon aléatoire à la recherche des fourmis à attaquer mais également une certaine quantité de nourriture et de matériaux. Pour la réalisation du projet, nous avons choisi de représenter chaque entité par une classe. Nous mettons à l'œuvre des principes de génie logiciel en programmation objet, en particulier la modularité, l'abstraction, l'encapsulation, le polymorphisme et la documentation.

Méthode et Outils

Notre démarche pour ce projet consiste à utiliser des structures de données comme des listes de type 'Vector', afin éviter les exceptions de modification qui peuvent survenir. Nous utilisons aussi des tableaux et des variables pour le stockage d'information. Le langage choisi pour la réalisation du projet est Java. Pour la réalisation du projet nous avons utiliser les librairies suivantes:

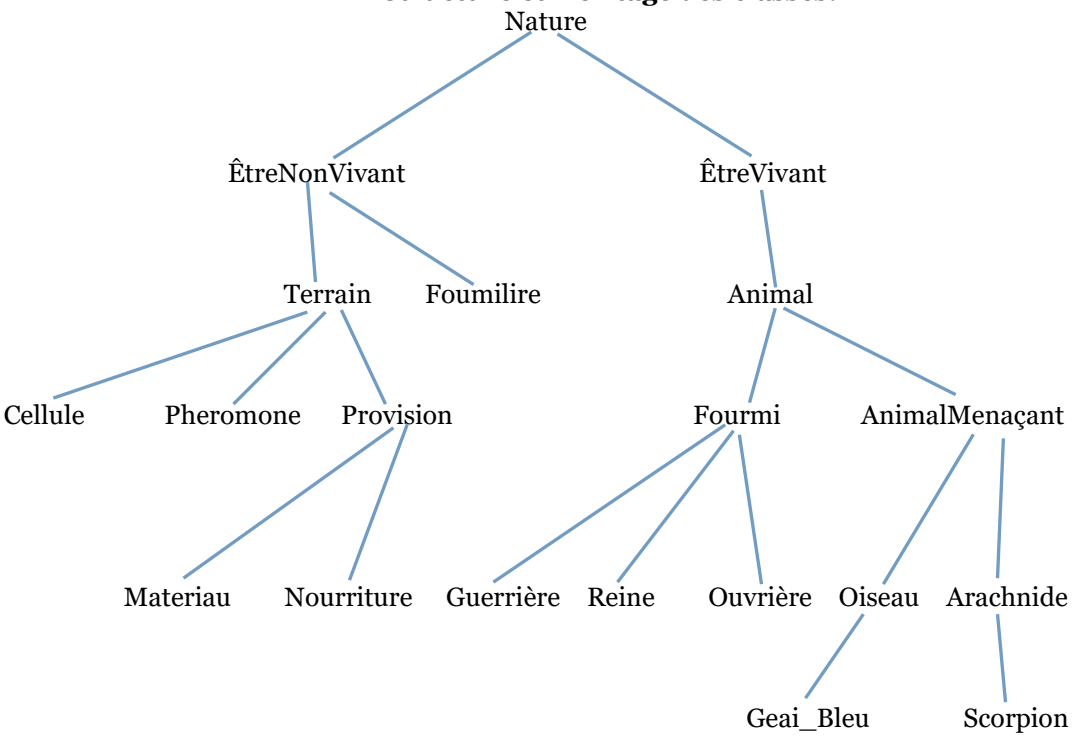
java.awt.Shape
java.awt.Point
java.awt.Color
java.awt.Rectangle
java.util.Vector
java.util.Random
java.awt.*
javax.swing.ImageIcon
java.util.Iterator
javax.swing.ImageIcon

Structure du programme

Nous représentons chaque entité par sa propre classe et chaque entité possède aussi une image pour sa représentation visuelle. Dans la classe 'GUI_for_Displayable', nous avons créer une liste des objets à afficher et puis grâce à l'objet de type 'Graphics' fournit dans cette classe, nous parcourons cette liste d'objet, en dessinant sur l'écran la représentation visuelle de chaque objet sur l'écran. Nous avons aussi une deuxième liste qui sert à garder de façon temporaire les objets à enlever de l'écran. Nous l'assemblage de tous les objets, nous avons du choisir une' classe dans laquelle les interactions entre les objets sont gérées sont la classe Terrain et la classe 'Fourmiliere'. Presque tous les objets possède, de façon directe ou indirecte, le même objet de type Terrain et celui de type 'GUI_for_Displayable', représentant la fenêtre sur laquelle tout se passe. La classe 'Fourmiliere' sert comme la classe de gestion de tous les objets de type 'Fourmis' qui en font partie.

La Terrain sert comme la classe de gestion de la totalité du projet, en gérant les interactions entre les fourmilières , les animaux menaçants, le terrain sur lequel la simulation se déroule et le temps.

Structure et Héritage des classes.



Interface:
Displayable: Elle contient les fonction getShape et getColor.
Habitude: Elle contient les fonctions ‘deplacer’ , ‘appelTimer’ et ‘manger’.

Description Des Classes

Classe Nature: Elle représente la classe dont tous les autres classes héritent. Elle contient des attributs de type Point qui représente les coordonnées x,y d’un objet sur le terrain, de type ‘Shape’ qui représente la forme géométrique d’un objet, de type ‘Color’ qui représente la couleur d’un objet. Son constructeur initialise les attributs de type ‘Int’ avec les dimensions de la fenêtre. Dans

la classe, nous trouvons des fonctions ‘getShape’ qui retourne un objet de type ‘Shape’ et ‘getColor’ qui retourne un objet de type ‘Color’.

Classe ÊtreNonVivant: Elle représente la classe dont tous les Êtres non vivant héritent, comme par exemple, la classe Terrain ou la classe Provision.

Classe ÊtreVivant: Elle représente la classe dont tous les Animaux héritent. Elle aussi représente aussi une idée abstraite d’un être vivant. Son constructeur prend en paramètres les coordonnées de l’être vivant sur le terrain.

Classe Terrain: Elle représente le terrain sur lequel tout se passe. Son constructeur prend en paramètre un point, une couleur et un objet de la fenêtre sur laquelle le terrain va être affiché. Elle contient une liste de type ‘Cellule’ contenant les cellules du terrain, une liste de type ‘AnimalMenaçant’ contenant les animaux menaçants sur le terrain et une liste de type “Fourmilière” qui contient les fourmilières du terrain. Dans la classe, nous trouvons les fonctions ‘getCellules’ qui créer et ajoute des cellules dans la liste de cellules du terrain, la fonction ‘retourneCellule’ qui retourne la cellule dont l’indice dans la liste de cellules du terrain correspond à la valeur passée en paramètre, la fonction ‘peupler’ qui remplit le terrain avec un certain nombre de fourmilières, la fonction ‘simulation’ qui déplace les animaux sur le terrain et gère les interactions entre eux, en d’autre terme, c’est la fonction qui fait tourner le programme, la fonction ‘miseAjour’ qui modifie le terrain, en ajoutant ou en enlevant des éléments en fonction des interactions entre tous les animaux, la fonction ‘nourritureEstDispo’ qui met quantité de la nourriture disponible sur le terrain et la fonction ‘verifierMortalite’ qui enlève les animaux menaçants qui sont morts du terrain et les remplace par de la nourriture. Cette classe nous permet de gérer les interactions entre les différents objets et éléments du programme.

Classe Animal: Elle représente une fourmis de façon abstraite. Son constructeur prend un objet de type Point, indiquant la position exacte de l’animal sur le terrain. Elle contient des attributs qui décrivent son état (vivant ou mort), son image, la direction sur le terrain. Elle contient la fonction ‘draw’ qui prend un paramètre un objet de type ‘Graphics’ permettant de dessiner l’animal sur le terrain et un objet de type ‘Component’ précisant la composante sur laquelle l’animal doit être dessiné, la fonction ‘addImage ‘qui ajoute l’animal dans la liste des objets à dessiner sur le terrain, les fonctions ‘enRegleX’ et ‘enRegleY ‘qui permettent de vérifier si l’animal se trouve entre les délimitations de la fenêtre/terrain, la fonction ‘mettreAJourEtat’ qui change l’état de l’animal lorsque l’animal meurt, la fonction ‘deplacer’ qui permet à l’animal de se déplacer sur le terrain de façon aléatoire, la fonction ‘appelTimer’ qui change la direction de l’animal et la fonction manger qui nourrit l’animal.

Classe Cellule: Elle représente la cellule d’un terrain. Son constructeur prend en paramètre des attributs de type ‘Point’, correspondant aux coordonnées de la cellule sur le terrain, de type ‘Shape’ correspondant à la forme géométrique de la cellule , de type ‘Color’ correspondant à la couleur de la cellule et l’objet de la fenêtre sur laquelle elle va être affichée. Elle contient la fonction ‘ajouterNourriture’ qui remplit le terrain d’une certaine quantité de nourritures et la fonction ‘ajouterMateriau’ qui remplit le terrain de matériaux.

Classe Pheromone: Elle représente la phéromone larguée par la fourmi ouvrière pendant sa recherche de nourriture. Son constructeur prend en paramètre les coordonnées de la phéromone, la couleur de la phéromone et la fenêtre sur laquelle la phéromone apparait. Elle contient la fonction ‘draw’ permettant de dessiner la phéromone sur l’écran, la fonction ‘addImage’ qui ajoute la phéromone dans la liste des objets à être affichés sur l”écran et la fonction ‘rmImage’ qui enlève la phéromone de l’écran.

Classe Provision: Elle représente les fournitures/outils dont les fourmis ont besoin, comme de la nourriture et du matériau. Son constructeur prend en paramètre les coordonnées de l’objet sur le terrain, la couleur de l’objet et la fenêtre sur laquelle nous affichons les provisions.

Classe Fourmilière: Elle représente une fourmilière contenant un certain nombre de fourmis. Son constructeur prend un objet du terrain sur lequel elle apparait, l'indice d'une cellule du terrain sur laquelle, elle va être positionnée et la fenêtre sur laquelle elle apparait. Elle comporte une liste de fourmis et une liste de provision, c'est-à-dire la liste de nourriture et de matériaux de la fourmilière. Nous y trouvons la fonction 'genesis' qui fait naitre les premières fourmis de la fourmilière et les ajoute dans la liste de fourmis de la fourmilière, la fonction 'nourritureRestant' qui retourne la quantité de nourriture disponible dans la fourmilière, la fonction 'positionnerFourmilière' qui positionne la fourmilière sur le terrain et plus précisément sur la cellule du terrain précisée en paramètres, la fonction 'retourneNourriture' qui retourne une portion de nourriture disponible dans la fourmilière, la fonction déplacer qui déplace les fourmis présentes dans la fourmilière, la fonction 'verifierMortalite' qui enlève une fourmi du terrain lorsqu'elle meurt, la fonction 'appelTimer' qui change la direction des fourmis présentes dans la fourmilière, la fonction 'TimerAlimentation' qui nourrit toutes les fourmis de la fourmilière, la fonction 'verifierEsperance' qui change l'état de vie des fourmis de la fourmilière une fois qu'elles ont dépassé leur durée de vie.

Classe Fourmi: Elle représente une fourmi de façon abstraite. Son constructeur prend en paramètre les coordonnées de la fourmi sur le terrain et la fourmilière à laquelle la fourmi appartient.

Classe AnimalMenaçant: Elle représente un animal menaçant de façon abstraite.

Classe Materiau: Elle représente l'outil dont les fourmis ont besoin pour la construction de leur nid. Son constructeur prend en paramètre les coordonnées du matériau sur le terrain et la fenêtre sur laquelle le terrain apparait.

Classe Nourriture: Elle représente la source d'alimentation des fourmis. Son constructeur prend en paramètre les coordonnées de la nourriture sur le terrain et la fenêtre sur laquelle elle apparait.

Classe Guerrière: Elle représente une fourmi guerrière. Son constructeur prend la fourmilière à laquelle elle appartient. Le moment de sa naissance, elle apparait à une position aléatoire dans sa fourmilière. Elle contient la fonction 'deplacer ' qui lui permet de se déplacer de façon aléatoire sur le terrain, la fonction 'detectCollision' qui lui permet de détecter une collision avec différents élément sur le terrain, la fonction attaquer, qui permet à la guerrière d'attaquer ses ennemis, la fonction 'changeDirection' qui change la direction de la guerrière, la fonction 'orientationImage' qui permet de changer l'orientation de l'image de la guerrière en fonction de sa direction, la fonction 'appelTimer' qui qui fait appel à la fonction 'changeDirection' et la fonction manger qui nourrit la guerrière. La guerrière attaque les animaux menaçants et les guerrières qui ne font pas partie de sa fourmilière.

Classe Reine: Elle représente la fourmi reine. Son constructeur prend la fourmilière à la elle appartient. Elle contient la fonction 'incuber' qui fait naître le type de fourmi passé en paramètre et l'ajoute dans la liste des fourmis de sa fourmilière, la fonction 'retourneQuantiteDeFourmi' qui retourne la le nombre de fourmis de ce type disponible dans sa fourmilière, la fonction 'ecloreTimer' qui vérifie s'il y a une pénurie d'un certain type de fourmis dans la fourmilière et si c'est le cas fait, appel à la fonction 'ecloreTimer' pour en produire, la fonction 'manger' qui nourrit la reine, la fonction 'deplacer' qui immobile la reine et ne lui permet pas de se déplacer et la fonction 'appelTimer' qui ne change pas la direction de la reine.

Classe Ouvrière: Elle représente une fourmi ouvrière. Son constructeur prend la fourmilière à laquelle l'ouvrière appartient. Le moment de sa naissance, elle apparait à une position aléatoire dans sa fourmilière. Elle contient une liste de phéromones, lesquelles, elle largue sur le terrain pendant sa recherche de nourriture. Elle contient la fonction 'deplacer' qui déplace la l'ouvrière sur le terrain de façon aléatoire, la fonction 'retourAuNid' qui indique à l'ouvrière le chemin de

retour à sa fourmilière et permet à l'ouvrière de ramener les bénéfices de sa recherche, grâce aux phéromones laissées sur le sol, la fonction 'detecteCollision' qui permet la détection de collision entre l'ouvrière et différents éléments du terrain, la fonction 'changeDirection' qui change la direction de l'ouvrière, la fonction 'orientationImage' qui change l'orientation de l'image de l'ouvrière, en fonction de sa direction, la fonction 'appelTimer' qui fait l'appel à la fonction 'changeDirection' et permet aussi à l'ouvrière de larguer une phéromone sur le sol et la fonction qui nourrit l'ouvrière et la fonction 'eliminerDoublant' qui la provision volée par une ouvrière appartenant à une autre fourmilière.

Classe Fourmilier: Elle représente un fourmilier de façon abstraite. Son constructeur prend des coordonnées du fourmilier sur le terrain.

Classe Oiseau: Elle représente un oiseau de façon abstraite. Son constructeur prend en paramètre les coordonnées de l'oiseau.

Classe Geai_Bleu: Elle représente un geai bleu. Son constructeur prend en paramètre le terrain sur lequel le geai bleu apparait. Elle contient la fonction 'deplacer' qui déplace le geai, la fonction détecteCollision détècte des collisions, la fonction 'creerMouvementImage' qui permet au geai bleu de voltiger sur le terrain, la fonction 'appelTimer' qui change la direction du geai bleu et la fonction 'manger' qui nourrit le geai.

Classe Arachnide: Elle représente un arachnide. Son constructeur prend en paramètre le terrain les coordonnées de l'arachnide.

Classe Scorpion: Elle représente une scorpion. Son constructeur prend en paramètre le terrain sur lequel le scorpion apparait. Elle contient la fonction 'deplacer' qui déplace le scorpion, la fonction 'détecterCollision' qui détecte des collisions, la fonction 'appelTimer' qui change la direction du scropion et la fonction 'manger' qui nourrit le scorpion.

Classe Pangoin:

Classe Horloge: Elle représente l'horloge du terrain. Elle sert à déclencher des évènements sur le terrain. Elle contient des attributs et des fonctions relatifs au temps. L'horloge est initialisée e à zéro, lors de sa création. Elle content la fonction 'demarrer' qui démarre l'horloge et la fonction 'afficher' qui affiche l'heure.

Conclusion

Nous avons eu quelques problèmes lors de l'utilisation des listes et plus précisément des exceptions dues à des modifications concurrentes. Nous avons décidé d'utiliser des images pour le projet, car nous nous sommes rendu compte que ça donnerait une représentation plus réaliste et ça nous donnerait plus d'envie d'aller au-dessus de ce qui était demandé. Ce qui nous a embêté pour la réalisation du projet, c'était le fait que l'interface fourni était plutôt destinée pour la représentation visuelle des objets par des formes géométriques et pas par des images, nous avons essayer de la changer et de l'adapter, de façon à que nous punissions représenter des objets par des images sur l'écran. Nous avons appris une pléthore de choses sur le fonctionnement de java et sur la programmation objet en général et nous avons également pu se familiariser avec la documentation de java en ligne qui, laquelle nous avons consulté tout au long de la réalisation du projet.