**INSTITUT UNIVERSITAIRE DE TECHNOLOGIE**de PAUL SABATIER-TOULOUSE III   
**Département GEII**115 Route de Narbonne,   
31077 Toulouse



Projet de programmation  
Simulateur de fourmis

Destinataires :

COMBETTES Stéphanie   
LAURENS Patrick

**Florent LABRUNE Année 2016-2017**

SOMMAIRE

[REMERCIEMENT 2](#_Toc485067630)

[INTRODUCTION 2](#_Toc485067631)

[L’entreprise JLOGICIELS 2](#_Toc485067632)

[I. Présentation de l’entreprise 2](#_Toc485067633)

[II. Activités de l’entreprise 2](#_Toc485067634)

[III. Organisation de l’entreprise 2](#_Toc485067635)

[Contexte du stage 2](#_Toc485067636)

[Analyse de la demande 2](#_Toc485067637)

[Environnements, démarche et choix techniques 2](#_Toc485067638)

[Conclusion 2](#_Toc485067639)

[A. Bilan professionnel 2](#_Toc485067640)

[B. Bilan personnel PPP 2](#_Toc485067641)

[Lexique 2](#_Toc485067642)

[Table des illustrations 2](#_Toc485067643)

[Annexe 2](#_Toc485067644)

[Résumé 2](#_Toc485067645)

[Abstract 2](#_Toc485067646)

# INTRODUCTION

Dans le cadre du DUT génie électrique et informatique industrielle les bases de la programmation logicielle nous sont enseignées dans les but de nous familiariser avec l’algorithmie .de base et de découvrir les bases du C, très utile dans le domaine de l’informatique industrielle et embarquée. Afin de simplifier les programmes, ce projet est réalisé dans un environnement C++, ainsi les notions de mémoire et de pointeur ne seront pas abordées lors de ce projet.

Le but de ce projet est de développer un programme qui simulera de façon basique une fourmilière. Cette simulation, paramétrable, devra reproduire certain comportements des fourmis et retranscrire ces comportements à l’utilisateur à travers un affichage simple mais clair. Tous ces comportements seront détaillés dans le cahier des charges présent dans ce rapport.

Dans un premier temps, nous verrons en détail le but du programme et toutes les subtilités qu’il doit intégrer. Par la suite nous nous intéresserons à la partie technique, dans cette partie nous verrons comment le programme est structuré et nous verrons en détail le fonctionnement d’une de ses parties. Enfin un bilan sera fait sur ce projet, ce qui a été réalisé, son déroulement et ce qu’il m’a apporté.

# Fonctionnalités et comportements du simulateur

## Fourmilière dans la vraie vie et dans le simulateur

Dans la nature les fourmis d’une fourmilière sont divisées en trois castes : la reine (unique pour une fourmilière dans la majorité des cas), les ouvrières et les mâles. Le but est de simuler les ouvrières et plus particulièrement celles chargées de ramener de la nourriture.

Ces ouvrières explorent les alentours de la fourmilière jusqu’à trouver une source de nourriture. Une fois celle-ci trouvée, la fourmi en ramène une partie laissant une trace faible et confuse de phéromone derrière elle afin d’indiquer aux autres le chemin vers cette source. Au fur et à mesure que les fourmis vont faire des allers-retours entre la fourmilière et cette source la trace va se renforcer jusqu’à créer une véritable autoroute de fourmis. Une fois la source épuisée les fourmis n’alimentent plus la trace de phéromone qui finit par disparaitre.



Figure 1 : fourmis suivant une trace de phéromone

Dans la simulation le comportement des fourmis sera sensiblement le même. Si une fourmi croise de la nourriture elle la ramènera à la fourmilière tout en laissant une trace de phéromone. Si une fourmi croise une trace elle la suivra. La seule différence avec la réalité est que dans la simulation la trace est claire dès le premier dépôt de phéromone.

## Cahier des charges

### La fourmilière

C’est la « maison » des fourmis. Lorsqu’une fourmi trouve de la nourriture c’est là qu’elle doit la ramener. C’est également de la que les fourmis apparaissent en début de simulation. Au cours de la simulation des fourmis peuvent naitre, c’est également là qu’elles apparaîtront. Toute la nourriture récoltée sera stockée dans la fourmilière. Lorsqu’une fourmi nait, le stock diminue de 50. Le processus de naissance ne peut s’enclencher que s’il y a de la nourriture stockée. La fourmilière sera signalée distinctivement sur l’affichage de la simulation.

### Les fourmis

Leur cycle de vie est simple, elles naissent à la fourmilière puis se déplace aléatoirement sur la carte. Ceci jusqu’à ce qu’elles croisent une trace de phéromone ou de la nourriture. Dans le premier cas elles suivront cette trace jusqu’au bout. Dans le deuxième elles rentreront à la fourmilière en laissant une trace ou en renforçant une trace déjà présente. Tout au long de leur vie les fourmis perdent aléatoirement des points de vies. Lorsqu’elles arrivent à 0, elles meurent et disparaissent de la simulation. Les fourmis seront signalées sur l’affichage par ‘|’ ou ‘\_’ en fonction de la direction dans laquelle elles se dirigent.

### Les traces de phéromone

Une trace va toujours d’une source de nourriture vers la fourmilière. Ainsi les fourmis la suivant n’ont qu’à juste la remonter pour se diriger vers la source de nourriture. Mais les traces ne sont pas infinies. Au fil du temps elles s’affaiblissent jusqu’à disparaitre. Une trace est donc divisé en plein de petits tronçons avec chacun une force. Lorsqu’un tronçon voit sa force tomber à 0 les fourmis ne pourront plus suivre la piste jusqu’au bout. Ainsi une fourmi suivant une trace et arrivant sur un tronçon avec une force de 0 arrêtera de la suivre et continuera ses déplacements aléatoirement. Chaque tronçon d’une trace est représenté par un ‘.’ Ou par un ‘i’ si il y a une fourmi ou plusieurs fourmis sur ce tronçon.

### Les sources

C’est là où se trouve la nourriture que les fourmis doivent récolter. Chaque source apparait avec une certaine quantité de nourriture. Chaque fois qu’une fourmi ramène de la nourriture, la source faiblie jusqu’à tomber à 0 et disparaitre. Chaque source est signalée par un ‘N’.

### L’affichage

C’est l’interface entre la simulation et l’utilisateur. Il permet à l’utilisateur d’avoir un rendu en temps réel de la simulation. Comme indiqué précédemment chaque élément de la simulation a sa représentation.

### Les modes de simulation

Le simulateur offre à l’utilisateur 2 modes :

* Le mode paramétré : l’utilisateur configure entièrement sa simulation, il choisit le nombre de fourmis en début de partie, leur vie de départ, le taux de naissance de la fourmilière, le nombre de source sur la map, la quantité dans une source et la force de base d’un tronçon de phéromone.
* Le mode automatique : la simulation se lance avec les paramètres par défaut, 20 fourmis, 200 points de vie, un taux de 0.25 (0 = pas de naissance, 1 = naissance en permanence), 10 sources, 500 de nourriture par source et force de 40 pour un tronçon.

## Scenario

### Choix du mode

L’utilisateur choisi le mode automatique ou manuel. S’il choisit le mode paramétré, un menu s’affiche dans lequel il peut choisir sa configuration (voir annexe 1). Sinon la simulation se lance avec les paramètres par défaut. A noter que si le joueur choisis le mode paramétré, les paramètres non configuré seront pris par défaut.

### Déroulement de la simulation

Durant toute la durée de la simulation les fourmis vont évoluées selon les règles défini dans le cahier des charges et en fonction de la configuration choisit par l’utilisateur.

### Fin de la simulation

La simulation se termine lorsqu’il n’y a plus de fourmis en vie et qu’il n’y a plus de nourriture dans la fourmilière ou que le taux de naissance soit paramétré sur 0 ce qui signifie qu’il ne peut plus y avoir de naissance de fourmi. Le programme affiche alors le nombre de tours qu’a duré la simulation.

# Structure et fonctionnement du simulateur

## Structure du simulateur

### Les structures

Les structures représentent toutes les données manipulées par le simulateur. Elles permettent de structurer les informations et de regrouper les informations en lien dans une même « boite ». Voici les structures que j’ai utilisées, les variables à l’intérieur et leur utilité :

|  |  |  |
| --- | --- | --- |
| **Nom de la structure** | **Ce qu’elle stock** | **Commentaire** |
| t\_coord | Coordonnée en x  Coordonnée en y | Représente les coordonnées d’un point sur la map |
| t\_fourmi | coord | Coordonnées de la fourmi |
| Nourriture | Quantité de nourriture transportée |
| Vie | Point de vie restant |
| Direction | Direction de la fourmi pour adapter l’affichage |
| idChemin | Id du chemin de phéromone suivi (-1 si aucun chemin suivi) |
| etat | Etat de la fourmi (voir zoom sur point technique) |
| t\_phero | coord  force | Force de la phéromone aux coordonnées coord |
| Chemin\_phero | Pheros | Tableau des phéromones qui composent le chemin |
| nbCoord | Nombre de phéromone (tronçon) du chemin |
| Id | Identifiant du chemin |
| fourmiliere | coord | Coordonnées |
| nourriture | Quantité de nourriture stockée |
| fourmis | Tableau des fourmis de la fourmilière |
| Nbfourmis | Nombre de fourmis dans la fourmilière |
| Source\_nourriture | Coord  Reste | Source de nourriture aux coordonnées coord |
| T\_simulation | maMap | Tableau a 2 dimensions de caractère représentant l’affichage de la simulation |
| Maison | Fourmilière de la simulation |
| Sources | Tableau des sources de nourriture |
| nbSources | Nombre de source de nourriture |
| Chemins | Tableau des chemins de phéromones |
| nbChemin | Nombre de chemins |

### Les variables globales

Les variables globales représentent la configuration du simulateur. Ainsi depuis n’importe où dans le programme je peux avoir accès aux paramètres du simulateur. Voici les variables globales utilisées par le simulateur et leur utilité :

|  |  |  |
| --- | --- | --- |
| **Nom de la variable** | **Fonction** | **Valeur par défaut** |
| Fourmis\_base | Nombre de fourmis au lancement du simulateur | 20 |
| Vie\_base | Points de vie d’une fourmi à son apparition | 200 |
| Taux | Probabilité qu’une nouvelle fourmi apparaisse à chaque tour | 0.25 |
| nbSources | Nombre de sources de nourriture en début de simulation | 10 |
| Force\_base | Force d’une phéromone à son apparition | 40 |
| Quantite\_nour | Quantité de nourriture stockée par une source au début de la simulation | 500 |

En plus de ces variables globales il y en une qui s’appelle idChemin, qui est incrémentée à chaque fois qu’un nouveau chemin est créé afin que tous les chemins est un id différent.

### Les constantes

Les constantes, comme son nom l’indique, sont des valeurs inchangeables qui souvent représentent les limites du programme. Voici les constantes du simulateur, leur utilité et leur valeur :

|  |  |  |
| --- | --- | --- |
| **Nom de la constante** | **Fonction** | **Valeur** |
| X et Y | Limite de la map | 100 et 50 |
| MAX\_fourmis | Nombre maximum de fourmis dans la simulation | 200 |
| MAX\_SOURCE | Nombre maximum de sources de nourriture dans la simulation | 50 |
| MAX\_CHEMIN | Nombre maximum de tronçons dans un chemin de phéromones | 100 |
| ET\_AVANCER\_ALEA | Différents états que peut prendre une fourmi (voir zoom sur point technique) | 1 |
| ET\_SUIVRE\_TRACE | 2 |
| ET\_RENTRER\_HOME | 3 |

## Fonctionnement du simulateur

Les sous programmes permettent de séparer le programme en plusieurs parties indépendantes. Chaque sous-programme à son utilité et sa fonction. Voici les principaux sous programmes du simulateur, leurs entrées et sorties et leur fonction :

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du sous-programme | Entrée(s) | Sortie(s)/retour | Fonction |
| getIndexChemin | Simu, id | Index | Retourne l’index du tableau où est stocké le chemin avec l’id « id » dans la simulation « simu » |
| addPheroChemin | coord | Chemin | Ajoute une phéromone aux coordonnées « coord » dans le chemin « chemin » |
| spawnfourmi |  | fourmil | Ajoute une nouvelle fourmie dans la fourmilière « fourmil » |
| supprimerSource | indexSource | Simu | Supprime la source à l’index « indexSource » de la simulation « simu » |
| supprimerfourmi | index | Fourmil | Supprime la fourmi à l’index « index » de la fourmilière « fourmil » |
| fatiguerfourmis |  | fourmil | Retire des points de vie a toutes les fourmis de la fourmilière « fourmil » |
| fatiguerChemins |  | simu | Retire de la force a tous les phéromones dans les chemins de la simulation « simu » |
| isOnPhero | Simu, coord, | Chemin, bool | Test si les coordonnées « coord » sont sur une phéromone. Retourne true et renvoie le chemin correspondant dans « chemin ». retourne false sinon. |
| isOnHome | Simu, coord | bool | Retourne true si « coord » correspond aux coordonnées de la fourmilière |
| isOnNourriture | Simu, coord | index\_nourriture, bool | Test si les coordonnées « coord » sont sur une source. Retourne true et renvoie l’index de la source dans « index\_nourriture ». retourne false sinon. |
| deplacerAleafourmi |  | fourmi | Déplace aléatoirement la fourmi « fourmi ». 1/5 chance de changer de direction, continue tout droit sinon |
| deplacerHomefourmi | simu | fourmi | Deplace la fourmi « fourmi » vers la fourmiliere |
|  |  |  |  |
| **Nom du sous-programme** | **Entrée(s)** | **Sortie(s)/retour** | **Fonction** |
| dropPhero |  | Simu, fourmi | Depose une nouvelle phéromone aux coordonnées de la fourmi « fourmi ». Si la fourmi est sur aucun chemin, un nouveau chemin est créé |
| evolutionEtat | simu | Fourmi, simu | Fais évoluer l’état de la fourmi en fonction de son état courant et de sa situation |
| majMap | simu | simu | Met à jour la map à afficher à l’utilisateur en fonction de l’état de la simulation « simu » |
| menuPrincipal |  | choix | Affiche le menu principal et retourne le choix fais |
| menuParametre |  | choix | Affiche le menu des paramètres et retourne le choix fais |
| parametreNbFourmis |  |  | Demande à l’utilisateur le nouveau paramètre pour le nombre de fourmis en début de simulation |
| parametreVieFourmis |  |  | Demande à l’utilisateur le nouveau paramètre pour la vie des fourmis |
| parametreTauxNaissance |  |  | Demande à l’utilisateur le nouveau paramètre pour le taux de natalité de la fourmilière |
| parametreNbSources |  |  | Demande à l’utilisateur le nouveau paramètre pour le nombre de source dans la simu |
| parametreQuantiteNour |  |  | Demande à l’utilisateur le nouveau paramètre pour la quantité de nourriture dans les sources |
| parametreForcePhero |  |  | Demande à l’utilisateur le nouveau paramètre pour la force des phéromones déposées par les fourmis |

## Zoom sur le fonctionnement des fourmis

### Description

Les fourmis ont plusieurs comportements bien définis en fonction de la situation dans laquelle elle se trouve. Afin de simplifier la gestion de cette évolution de comportement, les fourmis sont vues comme des machines à états par le simulateur. On peut distinguer 3 états :

* Etat 1 : déplacement aléatoire, la fourmi ne sait pas où elle va, elle n’a pas de nourriture à ramener, elle ne suit aucune trace ou chemin. La fourmi se déplace donc aléatoirement jusqu’à tomber soit sur de la nourriture, soit sur un chemin de phéromones.
* Etat 2 : suivre une trace, la fourmi a croisé un chemin de phéromones et le suit en direction de la nourriture. Ceci jusqu’à arriver au bout de la trace ou sur de la nourriture.
* Etat 3 : rentrer à la fourmilière avec de la nourriture, la fourmi à croiser de la nourriture, en a ramassée et la ramène à la fourmilière. En plus de rentrer elle laisse une trace de phéromones pour revenir à la source de nourriture et indiquer aux autres le chemin vers cette dernière.

### Diagramme d’état

finTrace

onNourriture

onHome &  
onPhero

onHome &  
 !onPhero

onNourriture

ETAT 2

ETAT 3

ETAT 1

### Algorithme simplifié

Algorithme simplifié du sous-programme « evolutionEtat » :

DEBUT  
 EVALUATION etat\_fourmi

CAS etat\_avancer\_aleatoirement :  
 SI fourmi est sur nourriture  
 fourmi ramasse nourriture  
 etat\_fourmi = etat\_rentrer  
 SI existe\_chemin  
 idChemin\_fourmi = idChemin  
 FIN SI  
 SINON SI fourmi est sur pheromone

idChemin\_fourmi = idChemin  
fourmi\_etat = etat\_suivre\_chemin

SINON

Deplacer\_aleatoirement(fourmi)

FIN SI

FIN CAS  
CAS etat\_rentrer :

SI fourmi est sur fourmiliere

Deposer\_nourriture  
SI id\_chemin\_fourmi < 0

Etat\_fourmi = etat\_avancer\_aleatoirement

SINON

Etat\_fourmi = etat\_suivre\_chemin

FIN SI

SINON

Deposer pheromone  
deplacer\_vers\_fourmiliere(fourmi)

FIN SI

FIN CAS

CAS etat\_suivre\_chemin :

SI fourmi sur nourriture

Ramasser nourriture  
etat\_fourmi = etat\_rentrer

SINON

SI trace continue

Suivre\_trace  
SINON

Etat\_fourmi = etat\_avancer\_aleatoirement

FIN SI

FIN SI

FIN CAS  
FIN EVALUATION

FIN

# Bilan et conclusion

## Bilan

### Le produit final

Le simulateur final rempli tous les points fixés par le cahier des charges et tout cela avec une structure propre claire et efficace. Mais quelques imperfections persistent et devraient être corrigées dans de futures versions.

Premièrement le calcul du chemin entre une fourmi et sa fourmilière est peu efficace, bug quelques fois faisant des détours injustifiés et ne permet pas le contournement d’obstacle.

Ensuite la simulation est réaffichée à chaque tour intégralement à l’utilisateur. Il serait plus efficace de réafficher uniquement les éléments qui ont changés par rapport au tour précédent. Le programme serait ainsi plus léger à faire tourner. Il serait également possible de rajouter quelques couleurs afin de rendre l’affichage un peu plus esthétique.

Enfin une saisie de données contrôlées a été implémentée lors du paramétrage du simulateur afin que l’utilisateur ne dépasse pas les bornes fixées par le programme. Mais ce contrôle n’assure pas la nature de ce que rentre l’utilisateur. Cela signifie que si l’utilisateur saisie une chaine de caractères alors qu’un nombre est attendu, le programme plate.

### Apports du projet

Malgré tout ce que j’ai déjà appris durant mes deux années de dut informatique ce projets m’a quand même apporté des choses.

Tout d’abord il m’a permis de replonger dans le C, un langage que j’avais peu utilisé depuis un moment. Ainsi il m’a rappelé certaine comme chose comme les structures mais il m’a surtout obligé à développer un programme sans utiliser la programmation orientée objet.

Tout le projet étant développé dans l’environnement de développement codeblocks, j’ai appris à me servir du debugger de ce dernier. Le debugger est un outil un peu complexe à prendre mais qui s’avère être un outil redoutable pour trouver les erreurs de fonctionnement dans un programme.

Enfin, à travers les fourmis, j’ai utilisé une machine à états qui est une façon de développer peu utiliser en dut informatique mais très utile pour l’informatique industrielle et embarquée. Etant la partie qui m’intéresse le plus en informatique, il est très important de manipuler ce type de structure.

### Analyse du projet

L’analyse d’un problème est la base du développeur. L’avantage d’un projet aussi large qu’un « simulateur » c’est que justement il nécessite beaucoup d’analyse, de réflexion et de méthodologie. Au début je me suis un peu lancé à l’aveugle et je me suis très vite rendu compte que je faisais fausse route. Comme tous les projets que j’ai déjà réalisé, il m’a rappelé que analyse est synonyme d’efficacité.

## Conclusion

Pour conclure, ce projet m’a permis de revoir les bases du C mais également des choses plus avancées comme les structures. Il m’a également rappelé qu’il n’y a pas que le développement orienté objet dans la vie mais également d’autres méthodes comme les machines à états.

Ce projet m’a également fait une bonne piqure de rappel sur l’importance de l’analyse d’un problème et de la rédaction de son cahier des charges.

# Annexes

## Annexe 1 : le menu utilisateur

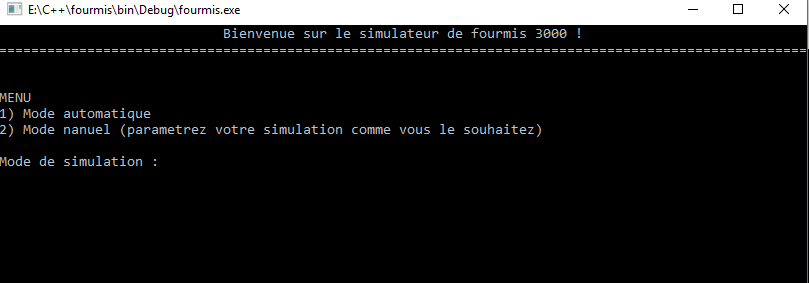


Figure 2 : menu général

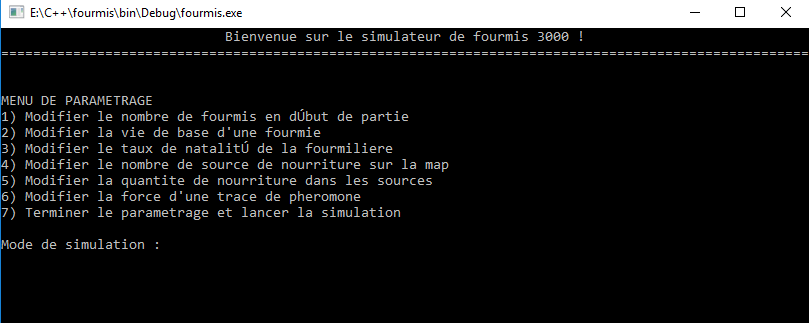


Figure 3 : menu de paramétrage

## Annexe 2 : Simulation en cours

## 

Affichage relatif a la simulation en cours

Fourmis

Fourmilière

Source de nourriture

Fourmi sur une phéromone

Chemin de phéromones