Compte-rendu de TP de programmation C/C++
Project Mourfiz

Fabien Steinmetz, Yannick Wurm

5 avril 2002

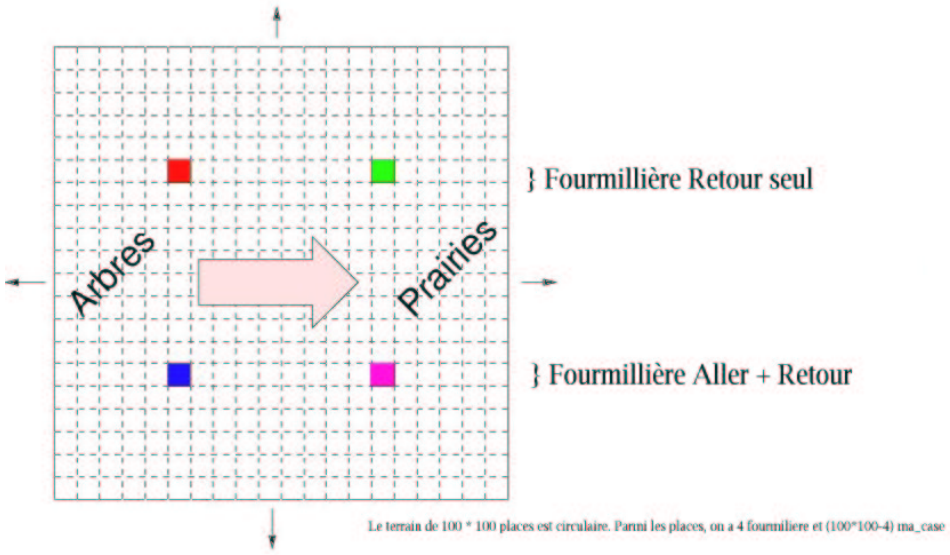
Résumé

Ce TP a été réalisé dans le cadre du cours de programmation de Jean-Michel Létang. À travers un petit projet ayant comme thème la simulation du comportement de fourmis, nous avons améliorés nos connaissances en matière de programmation orientée objet en C++.

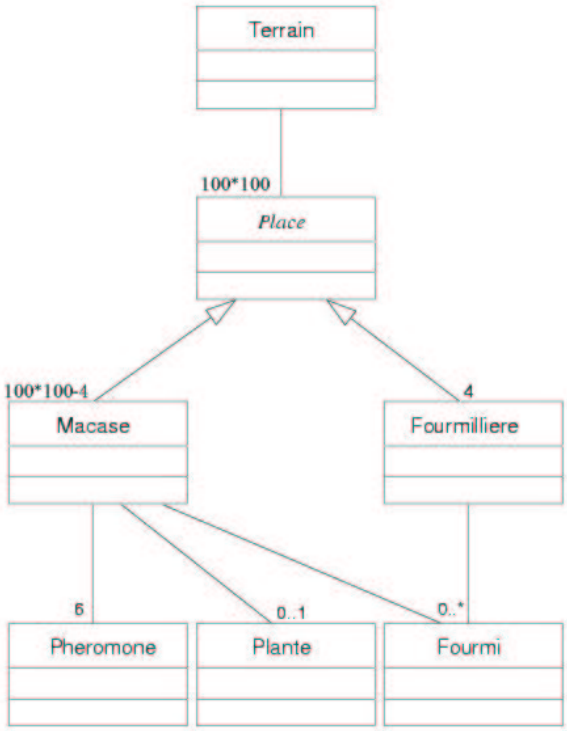
Table des matières

1 Descriptif

Le thème de ce projet est de créer un environnement permettant d'étudier le comportement d'automates simples. Nous avons choisi d'étudier le comportement de fourmis ouvrières, dont l'unique tâche est d'aller chercher de la nourriture et de la rapporter a leur fourmilière. Les fourmis ne sont pas orientées et évoluent sur un gridworld. Elles connaissent les informations suivantes concernant leur environnement : pour chacune des 4 places (les cases de notre gridworld) directement adjacentes à la leur, elles peuvent déceler la présence ou non d'une plante (source de nourriture), de leur fourmilière, ainsi que la présence des phéromones propres à leur fourmilière. En utilisant ces informations ainsi que l'intention de la fourmi d'aller chercher de la nourriture ou bien de rentrer à la fourmilière, on calcule une pour chaque place ; le choix de la place retenue pour son déplacement se fait par une lotterie biaisée.
Notre but est de mettre en évidence l'établissement d'une intelligence locale dans le comportent qu'adoptent les fourmis pour aller chercher de la nourriture. Pour cela, on travaille simultanément avec quatre situations différentes : On prend deux types de fourmis : lorsqu'elles partent de leur fourmilière pour chercher de la nourriture, toutes posent des phéromones dites de type retour, afin de pouvoir retracer leur chemin. En plus, la moitié des fourmis dépose lorsqu'elle rapporte de la nourriture à sa fourmilière un second type de phéromone, dit "aller". Une fourmi de ce type partant à la recherche de nourriture devrait alors suivre la trace des phéromones "aller". Nous testons nos deux types de fourmis chacune dans un environnement de type prairie (on y trouve des petites plantes distribuées uniformément) et aussi dans un environnement de type forêt où les plantes sont grandes et éparpillées.



Voici un schéma indiquant les différentes classes implémentées :



2 Spécificités d'implémentation

Pendant la phase de codage du programme, nous nous sommes donné du mal pour réflechir de manière orientée objet. Ainsi nous avons privilégié l'utilisation de l'héritage et de fonctions virtuelles à travers la classe virtuelle `place` et ses sous-classes `macase` et `fourmiliere`. Nous avons aussi essayé de rendre notre code le plus lisible possible, en nous mettant d'accord sur une nomenclature commune et généralisée ainsi qu'un style d'écriture commun. Ainsi on peut remarquer par exemple que toutes nos classes ont une fonction `afficher()` ainsi qu'une fonction `maj()`.

Afin d'avoir une bonne maintenabilité du code, nous avons aussi centralisés l'ensemble des constantes utilisés dans un ficher constantes.h

2.1 Liste chaînée de phéromones

Contrairement à ce qui avait été précisé dans l'enoncé, nous avons choisi de ne pas faire de tableau statique de phéromones. L'utilisation d'un tel tableau ne nous semblait pas adéquat pour plusieurs raisons. Tout d'abord, il faudrait se mettre d'accord sur les correspondances entre le type d'une pheromone (son sens ainsi que le type de fourmi par laquelle elle fut déposée) et sa position dans le tableau. Cela se traduit par une mauvaise lisibilité du code (par exemple quand on réfère aux phéromones à partir d'une place). On peut se dire alors qu'on stockerait le type de la fourmi et son sens dans des variables de la classe phéromone. Dans ce cas il semble illogique de vouloir stocker les phéromones dans un tableau avec une position fonction de leur type et de leur sens. C'est pourquoi nous avons opté pour l'utilisation de listes chaînées de phéromones dans les places (nous avons pour cela été obligés de rajouter plusieurs fonctions à la classe `place`).

2.2 Raccourcis clavier

Touche	Fonction
Esc	quitte le programme
²	passe a l'affichage par défaut (vue du dessus du terrain)
c	bascule l'affichage des cases du terrain
f	bascule l'affichage des fourmis
r	bascule l'affichage des fourmilières
t	bascule l'affichage des plantes
p	bascule l'affichage des pheromones
w	oriente la caméra (l'oeil de l'observateur) vers la gauche
x	oriente la caméra vers la droite
s	sauvegarde l'état de mourfiz dans un fichier
l	charge un état de mourfiz a partir du disque dur
+	affiche tous les éléments du terrain
-	n'affiche aucun élément du terrain
F1	Arrete ou Continue la croissance des plantes
UP	Avance la caméra en gardant la meme direction de vision
DOWN	Recule la caméra en gardant la meme diretction de vision
LEFT	La caméra se déplace vers la gauche en gardant la même direction de vison
RIGHT	La caméra se déplace vers la droite en gardant la meme direction de vison
PAGEUP	La caméra monte, tout en gardant le même point de visée.
PAGEDOWN	La caméra desend, tout en gardant le même point de visée.
BEGIN	On bascule à un affichage plus tridimensionnel

3 Difficultés rencontrées

A travers le réalisation du projet, nous nous sommes rendus compte de l'utilité du `printf()` quand il faut trouver la cause d'une erreur, notamment des erreurs de segmentation. Une des causes d'erreurs de segmentation inexplicables que nous avons identifiés sont les dépassements de limites de tableau.
Aussi avons-nous remarqués l'importance de la synchronisation des horloges lorsque l'on travaille sur plusieurs ordinateurs : en effet quand les horloges ne sont pas synchronisées, la récompilation par l'execution de make peut être uniquement partielle (on a pris le réflèxe, dès qu'on a une erreur inexplicable, de faire d'abord un `make clean` avant de relancer `make`).

3.1 Affichage

Nous n'avons pas trouvé de moyen simple permettant de basculer entre un deroulement du programme dans la console, sans affichage graphique, et l'utilisation de glut. En effet, en tant que telle, la fonction glutMainLoop() ne renvoie jamais la main au programme qui l'appelle (pour s'arrêter, elle utilise la commande `exit()` et non la commande `return`). C'est l'une des raisons pour laquelle nous avons decidé d'implémenter un contrôle par clavier sur l'affichage. Voici un tableau récapitulatif des commandes principales :

3.2 Enregistrement

Nous avons essayés d'implementer l'enregistrement de l'ensemble des données du programme, pour pouvoir les recharger en mémoire à un autre moment. (nous n'avons pas réussi - une erreur de ségmentation se produit à la lecture)
La stratégie était la suivante : étant donné que les nombres de fourmis, de phéromones, de plantes d'une case n'est pas fixe, on ne peut pas simplement faire des fwrite(fourmis_data , sizeof(objet), nombre d'objets) puis des `fread()` car lors de la lecture, on ne connait pas leur

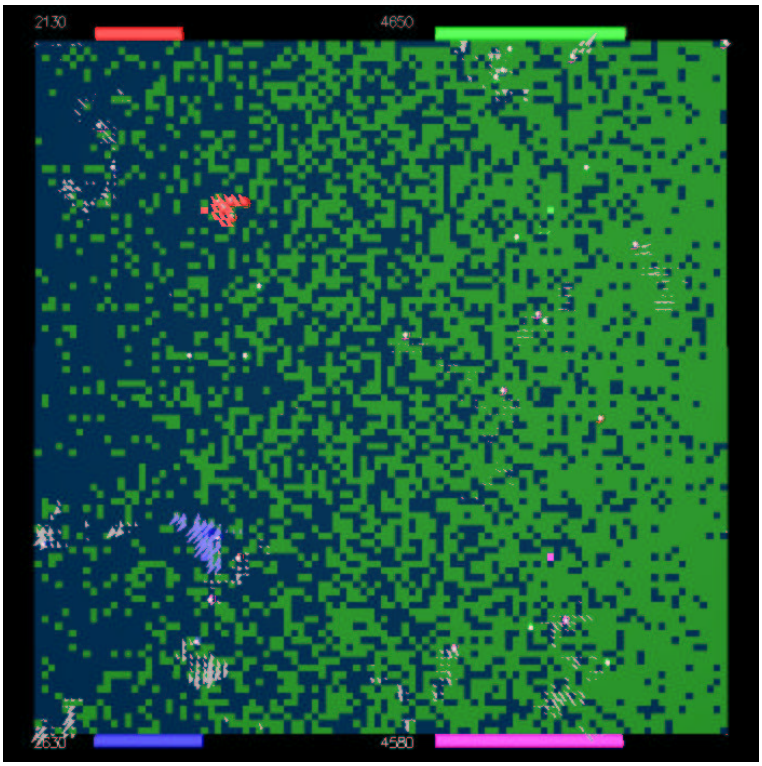
nombre. Pour remédier à cela, nous avons voulu créer préalablement un tableau regroupant ces donnés quantitatives pour chaque case. A partir de la lecture des données de ce tableau `etats_terrain`, on connaitraît, pour chaque place, le nombre d'éléments devant être lus.

3.3 Listes chaînées

Maintenant on maîtrise.

4 Observation du comportement des fourmis

Seules, nos fourmis ne sont pas intelligentes : les phéromones qu'elles déposent perdent rapidement leur puissance. On le voit sur la capture d'écran ci-dessous où l'on a 10 fourmis par fourmilière : les fourmis sont perdues, elles ne retrouvent pas leur fourmilière.

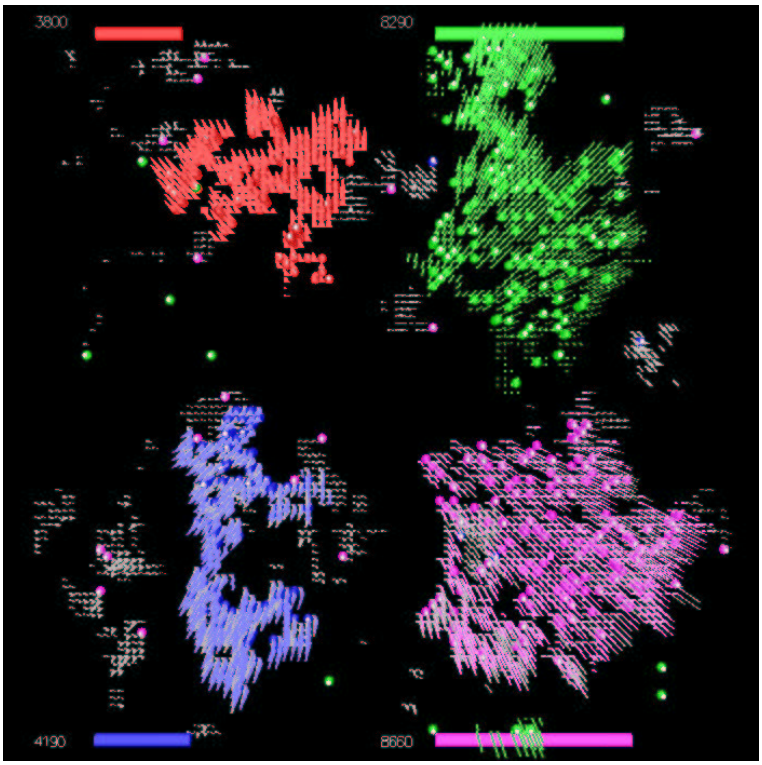


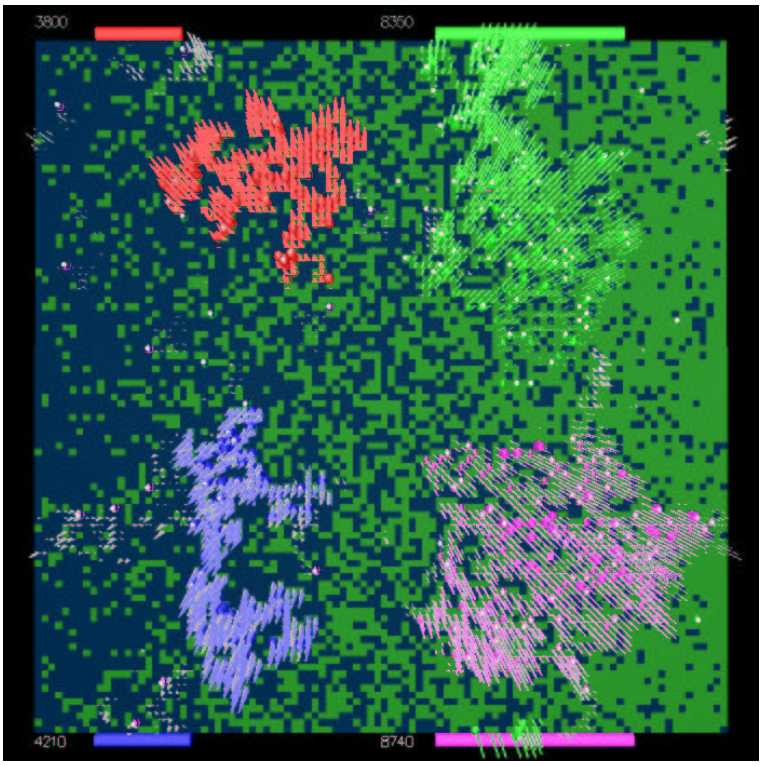
Nous avons constaté que lorsqu'une fourmi veut rapporter de la nourriture à sa fourmilière et qu'elle se rapproche de celle-ci, elle se retrouve prise dans un champ de phéromones de très forte intensité. La probabilité qu'elle choisisse de retourner sur la place qu'est sa fourmilière est alors pratiquement identique à celle d'aller sur la case d'à côté. Ainsi, nous nous sommes souvent retrouvés avec un groupe de fourmis tournant autour de leur fourmilière sans pour autant y rentrer. Afin de réduire ce type de comportement, nous avons décidé de rajouter aux fourmis une capacité supplémentaire : la reconnaissance de leur propre fourmilière. En rajoutant quelques lignes à la fonction fitness on augmente donc la probabilité qu'a une fourmi portant de la nourriture d'aller sur une place si celle-ci est sa fourmilière.

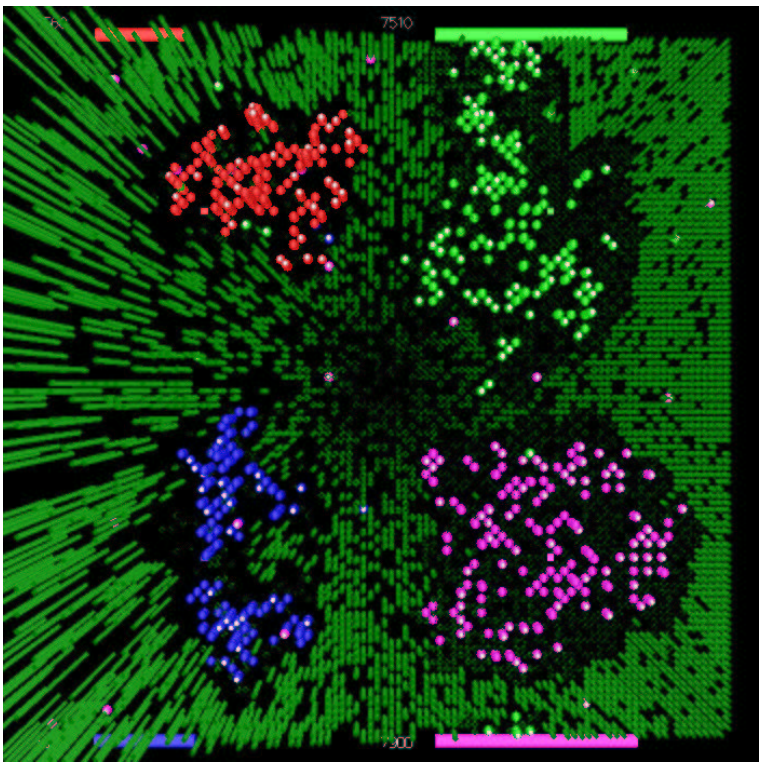
Cette modification nous donne un meilleur comportement, mais le résultat n'est pas toujours très satisfaisant - d'autant plus qu'il n'a d'effet que pour les fourmis voulant retourner à

leur fourmilière. Il persiste toujours une accumulation de fourmis à proximité de la fourmilière. Pour combler ce défaut (en effet, on voit rarement un groupe de fourmis tournant éperdument autour de leur fourmilière), on pourrait envisager de donner un meilleur sens de l'orientation à nos automates.
Une première façon de procéder serait de leur donner une indication sur la distance et la direction de leur fourmilière, et de tenir compte de ce facteur dans le calcul de la fitness d'une place (suivant que les fourmis portent ou cherchent de la nourriture).
Une autre possibilité, plus simple à implémenter, serait de donner aux fourmis la capacité de se souvenir de la dernière case sur laquelle elles étaient, et de faire en sorte que la probabilité qu'elles y retournent soit très faible (sauf évidemment si entre temps la fourmi a ramassé ou déposé de la nourriture et change de but). Les déplacements sembleraient alors certainement plus cohérents car plus directionnels.

A l'aide des différentes captures d'écran, il est possible de voir et d'expliquer le comportement de nos fourmis suivant le type de végétation qu'elles ont autour d'elles, et leur capacité à déposer des phéromones Aller.



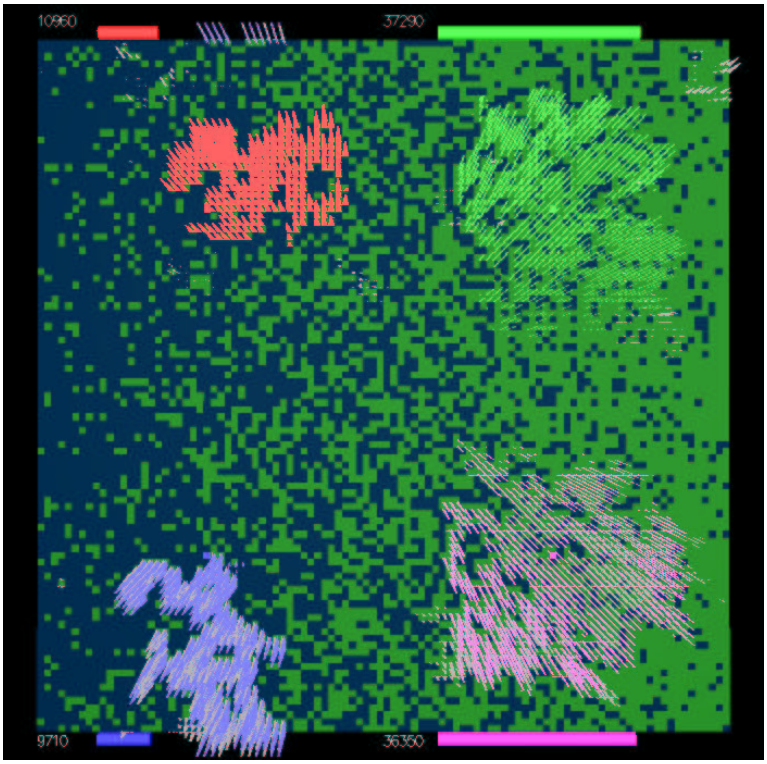




Cas des fourmis rouges (Phéromone retour + Végétation de type Arbre)
Elles sont assez proches de leur fourmilière de manière générale. Elles ramènent assez peu de nourriture par rapport aux autres fourmis de notre terrain : ce sont les moins efficaces.

Cas des fourmis Bleues (Phéromone A/R + Végétation de type arbre)
Ces fourmis sont un peu plus efficaces que les fourmis vertes vues précédemment. Elles sont

aussi très proche de leur fourmilière. On peut aussi remarquer que ces fourmis peuvent être un peu deportées vers la droite par rapport à leur fourmilière. Ceci serait dû au fait que beaucoup d'entre elles sont autour d'un meme point "d'approvisionnement" autour duquel il y a donc beaucoup de phéromones (Aller et Retour) ; les fourmis tendent à y rester.



Ces deux premiers types de fourmis se perdent en fait très rarement. Etant donné que la nourriture est très dense en certains endroits du terrain, elles ont tendence à aller toutes vers cet endroit. Le retour à la fourmilière semble poser problème. Les fourmis bleues sont plus efficaces du fait de la présence de phéromones Aller (en plus de la phéromone Retour qu'elles déposent toutes les deux).

Cas des fourmis Vertes (Phéromone Retour + Végétation de type Prairie)
Ce sont les plus efficaces de notre terrain (en faisant beaucoup d'observations et en laissant tourner le programme assez longtemps).

Cas des fourmis Violettes (Phéromone A/R + Végétation de type Prairie)
Elles sont un peu moins efficaces que les fourmis vertes, ce qui est assez surprenant, puisqu'on s'attendrait plutôt à ce qu'elles soient encore plus efficaces que les fourmis vertes, mais ce n'est pas le cas. On se rend compte de ce comportement surtout à long terme. Au début, elles sont en général les plus efficaces.
Explication : La végétation est de type prairie. D'autre part, elles déposent des phéromones aller. Ainsi quand elles auront à manger, elles déposeront des phéromones 'aller' puis reviennent à la fourmilière en suivant les phéromones 'retour'. Les fourmis sans nourriture à proxité des phéromones Aller qui viennent d'être déposées vont alors suivre les pheromones de notre fourmi ayant trouvé de la nourriture. Elles s'eloignent alors encore plus de la fourmilière puisque les réserves proches de la fourmilière ont déjà été exploitées. Or les phéromones ont une durée de vie limitée. Ainsi si les fourmis se sont aventurées trop loin, elles ne pourront plus suivre les

phéromones - qui auront alors disparu au cours du temps - pour pouvoir retourner dans leur fourmilière.

En fait, leur efficacité (vertes et violettes) est vraiment très proche.

Exemple de valeur prises (pour la qté de nourriture) à un instant donné :

Rouges	38 270
Vertes	46 170
Bleues	131 280
Violettes	129 800

Par ailleurs, on note que les fourmis sur végétation de type prairie ont beaucoup plus tendance à se perdre totalement sur le terrain alors que les fourmis sur végétation de type arbre se perdent beaucoup plus rarement. Ceci est en fait plutôt dû a l'étalement de petites quantités de nourriture plus rapidement exploitées dans le cas de celles qui se dispersent beaucoup. Mais la durée de vie de la phéromone a bien évidemment aussi un rôle dans cela.

5 Conclusion

En faisant ce TP, on a abordé la programmation en C++ d'une façon assez ludique par l'utilisation de l'OpenGL. Il nous a été très bénéfique puisqu'on a beaucoup cherché par nous-même les fonctions dont on a besoin, et les façons de résoudre certains problèmes que nous nous sommes posés au cours du dévelopement. On a vraiment cherché comment on pouvait faire pour implémenter telle fonctionnalité dans notre programme.
Le comportement des fourmis importe peu dans ce projet, puisque son objectif était plutôt de nous faire programmer et manipuler certaines librairies et certaines techniques pouvant se révéler très utiles.

Evidemment, s'il l'on voulait faire quelque chose de parfait, nous n'aurions jamais fini de retoucher, changer, modifier, améliorer notre programme. L'heure est venue de boucler le projet, un projet étant par définition toujours limité dans le temps...