

Rapport Globulation 2  
**Communication entre globules**

EPITA - Octobre 2004

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Principe</b>	<b>4</b>
2.1	Communication indirecte (phéromones)	4
2.1.1	Types de phéromones	4
2.1.2	Diffusion de phéromones	4
2.1.3	Influence sur le comportement	4
2.1.4	Caractéristiques des phéromones	5
<b>3</b>	<b>Implémentation</b>	<b>6</b>
3.1	Description de l'existant	6
3.1.1	Les paliers	6
3.1.2	Le prise de décision des unités	6
3.2	Phéromones	7
3.2.1	Modélisation	7
3.2.2	Ajout	7
3.2.3	Diffusion	8
3.2.4	Influence	8
3.2.5	Optimisation	9
3.2.6	Affichage	9
<b>4</b>	<b>Listing des modifications apportées</b>	<b>11</b>
<b>5</b>	<b>Bilan</b>	<b>12</b>
5.1	Difficultés rencontrées	12
5.2	Ajouts possibles	12

# 1 Introduction

Ce rapport a pour objectif de montrer les évolutions apportées par notre groupe au projet Globulation 2. Cet apport a été réalisé pour EPITA pour appliquer le cours d'Intelligence artificielle et Mondes virtuels dans les Jeux vidéos.

Globulation 2 est un jeu de Stratégie en Temps Réel innovant, qui minimise le micro-management en assignant automatiquement les tâches aux unités. Le joueur à juste besoin de choisir la quantité d'unités qu'il désire pour effectuer différentes tâches et les unités font leur possible pour satisfaire ces requêtes. Cela permet de gérer plus d'unités et de se focaliser sur la stratégie.

Globulation 2 est un logiciel libre, disponible sous les termes de la license publique générale GNU : GNU General Public License. Le code que nous fournissons est soumis aux mêmes termes.

Nous nous sommes intéressé particulièrement à la communication entre les globules. Nous avons envisagé deux approches :

- Communication indirecte par un système de phéromones.
- Communication directe par messages.

Dans ce rapport nous allons uniquement nous concentrer sur la communication indirecte.

D'abord nous allons présenter les motivations et les principes théoriques des ajouts que nous proposons. Ensuite nous verrons les modifications qui ont été nécessaires, en présentant à chaque fois le contexte actuel et comment nous nous y insérons.

## 2 Principe

### 2.1 Communication indirecte (phéromones)

Les phéromones sont des sortent d'odeurs qui peuvent être reconnues par les globules afin de s'échanger des informations locales.

Ainsi les globules peuvent :

- diffuser des phéromones ;
- être sensible aux phéromones présentes dans l'air.

Les globules vont donc se déplacer en laissant derrière eux des phéromones, récupérer les phéromones dans l'air et les diffuser à leur tour.

#### 2.1.1 Types de phéromones

Les différents types de phéromones sont :

**Passage** pour indiquer le passage d'un globule à cet endroit. Cela doit permettre des regroupements de globules et des trajectoires optimales. De plus si on veut protéger un endroit il est plus facile de n'avoir qu'un chemin à protéger que toute une zone.

**Danger** pour indiquer qu'une attaque a eu lieu à cet endroit ou que des ennemis ont été rencontrés. Les globules n'ont pas envie de se trouver dans ces endroits <sup>1</sup>.

**Mort** pour indiquer qu'un globule est mort à cet endroit, ce qui attriste notre globule. Le globule se recueille pour pleurer la mort de sont compatriote avant de repartir.

**Sexuel** pour indiquer si un globule est "en chaleur". Cela est inutile dans la version actuelle du jeu.

#### 2.1.2 Diffusion de phéromones

Les différents cas de diffusion sont :

- Diffusion des phéromones de **Danger** quand un bâtiment ou une unité est attaquée. Le sol, l'unité touchée et l'unité attaquante sont affectés par les phéromones.
- Diffusions des phéromones de **Mort** quand une unité est morte, globule ou bâtiment.
- Diffusion des phéromones de **Passage** quand un globule a une destination précise.
- Les phéromones **Sexuelles** ne sont pas diffusées.

#### 2.1.3 Influence sur le comportement

Les différentes classes de globules (guerriers, explorateurs et ouvriers) sont influencés différemment par les phéromones, ils réagiront a certaines plutôt que d'autres.

Voici le détails des différents comportements :

- Les **ouvriers** *suivent* les routes pour aller vers les ressources. Cela est surtout utile pour les longues distances où la route optimale va se former.
- Les **ouvriers** et les **explorateurs** *évitent* le danger.
- Les **explorateurs** *évitent* les routes.

<sup>1</sup>possibilité de coopérer avec le groupe 6 : zones dangereuses

- Les **guerriers** vont *vers* le danger quand ils sont en forme mais l'*évitent* quand ils sont blessés.
- **Tous** les globules se recueillent autour des morts si il ne se passe rien autour d'eux.

#### 2.1.4 Caractéristiques des phéromones

Il est important de noter que les phéromones sont :

- d'intensité variable afin de pouvoir jauger l'importance et la véracité de l'information apportée.
- volatiles et donc s'estompent avec le temps (l'intensité baisse jusqu'à être nulle).

Les phéromones n'appartiennent à aucune équipe, c'est-à-dire qu'elles ont la même influence sur toutes les équipes. Ainsi toutes les équipes vont suivre les mêmes routes et se rencontrer dans les points dangereux, tout comme tout le monde se recueillera sur les tombes.

## 3 Implémentation

### 3.1 Description de l'existant

Nous nous sommes basé sur la release alpha 8.9 pour faire nos modifications.

Nous allons présenter notre compréhension des deux parties que nous avons du modifier. La compréhension de ces parties à été essentielle pour la réalisation du projet.

#### 3.1.1 Les paliers

Le jeu avance par paliers, c'est-à-dire qu'il y a des sortes de tours dans lesquels les actions des joueurs, de l'intelligence artificielle et des unités est mise à jour. Puis le moteur graphique redessine tout.

On pourra trouver dans la doc<sup>2</sup> une brève description du graphe d'appel de cette procédure clé. Voici une version épurée des enchaînements des appels :

```
Engine.run();
  NetGame.step();
    GameGUI.step();
      Game.step();
        Map.step();
          Team[32].step();
            Unit[1024].step;
              Building[1024].step();
```

Pour nos modifications on se positionnera :

- au niveau des unités pour ce qui est de la création des phéromones et des messages.
- au niveau du jeu (Game) pour ce qui est de la mise à jour des phéromones et de l'affichage.
- au niveau de l'interface graphique (GameGUI) pour ce qui est de la mise à jour de l'interface.

#### 3.1.2 Le prise de décision des unités

Pour décider les unités passent par une suite de fonctions de prise de décision. La prise de décision est découpée en plusieurs étapes :

1. Vérification si on a besoin de se soigner et mise à jour de l'action et des données en conséquence.
2. Choix de l'activité à effectuer. Selon que l'unité est occupée ou pas, qu'il y a des tâches à faire, des upgrades possibles, qu'il soit blessé ou non, le globule va décider d'une activité.
3. Choix du déplacement selon son mode de déplacement et de son objectif.
4. Détermination du mouvement concret réaliser.
5. Execution réelle de l'action nécessaire.

<sup>2</sup>sur le CVS dans *doc/sourceCodeUnderstanding.txt*

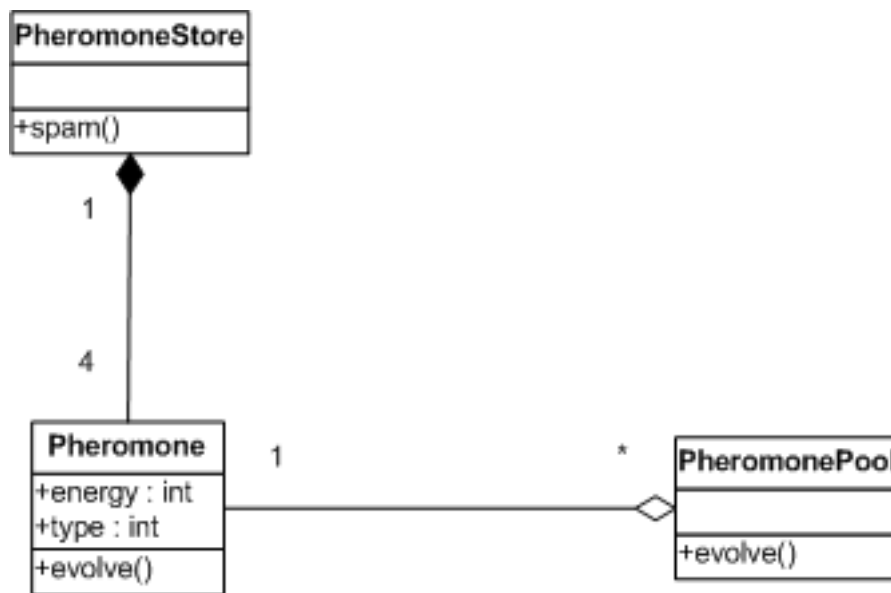


FIG. 1 – Diagramme de classe du module Pheromone.

## 3.2 Phéromones

### 3.2.1 Modélisation

Pour l'implémentation des phéromones nous avons rajouté deux fichiers : *Pheromones.cpp* et *Pheromones.h* contenant notre module de phéromones.

La classe `Pheromone` représente une phéromone, celle-ci est caractérisée par deux éléments :

- son énergie, c'est à dire le poids d'une phéromone. ;
- son type : Passage, Danger, Mort ou Sexuel.

Pour le typage nous aurions pu faire une hiérarchie de classe plutôt que de mettre une variable de type dans une unique classe, mais cela alourdi beaucoup le code et ralenti l'exécution car nous avons souvent besoin de connaître le type d'une phéromone depuis l'extérieur : nous ne pouvons pas nous abstraire.

Un `PheromoneStore` contient les quatre types de phéromone. C'est notre conteneur qui est stocké dans chacune des cases de la carte et par les unités. Il implémente une fonction de `spam()` qui sert à la diffusion des phéromones (voir partie sur la Diffusion 3.2.3).

Nous avons aussi un `PheromonePool` qui contient toutes les phéromones avec une énergie positive (voir la partie Optimisation 3.2.5).

### 3.2.2 Ajout

L'ajout de phéromone est fait à différents points correspondant à chaque cas de figure qui nous intéressent (pour les connaître voir 2.1.2).

Pour ajouter des phéromones à un endroit ou sur une unité on a juste à augmenter l'énergie de la phéromone voulue. En effet toutes les cases et toutes les unités contiennent les quatre phéromones en permanence y compris celle qui sont inactive avec une énergie de 0.

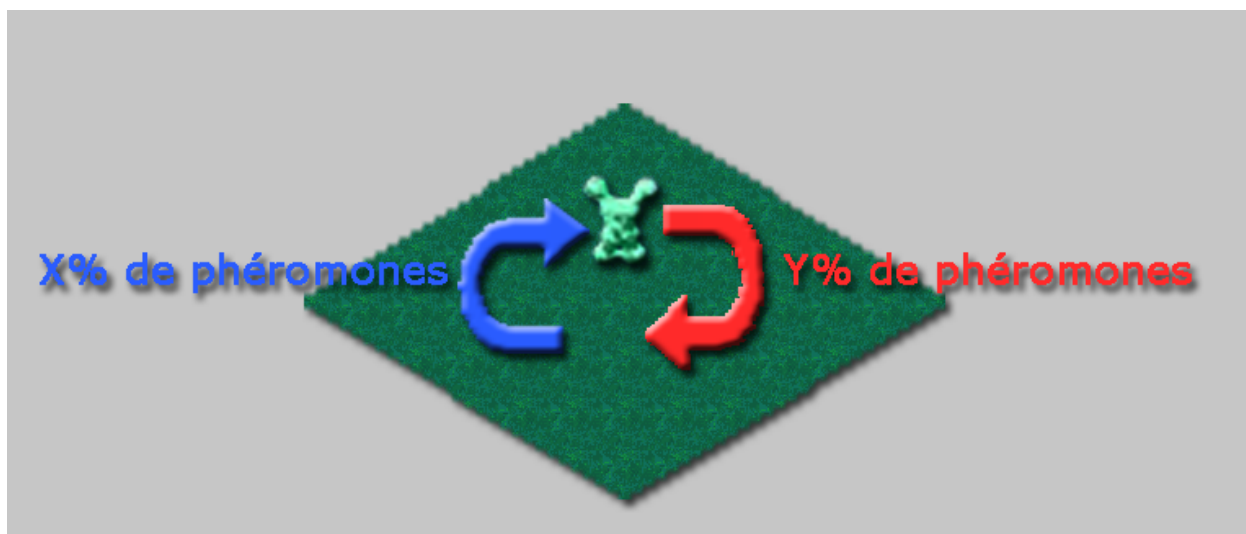


FIG. 2 – Diffusion des pheromones.

### 3.2.3 Diffusion

A chaque tour les phéromones évoluent, c'est-à-dire que toutes les énergies de toutes les phéromones sont mises à jour. C'est le rôle des fonctions `evolve()`. La formule d'évolution est une suite géométrique de la forme :

$$\text{energy} = 0.8 * \text{energy}; \quad (1)$$

La diffusion des phéromones se fait par les unités. Celles-ci en se déplaçant prennent les phéromones des cases qu'elles traversent, mais en même temps elles laissent un peu de leurs phéromones sur la case où elles sont. C'est la fonction `spam()`.

Il faut faire attention de ne pas réaliser les mises à jour séquentiellement mais bien en même temps.

### 3.2.4 Influence

A chaque tour de jeu, chaque entité de Globulation doit choisir une action à effectuer. La plupart de ces actions sont dictées par les règles présentes dans le code. (Par exemple, pour un explorateur : "visiter une case cachée"). Ces règles fonctionnent relativement bien. Lorsque l'entité ne trouve pas d'action à effectuer, elle se déplace aléatoirement. Nous avons modifié ces fonctions de manière à ce que le déplacement ne soit plus aléatoire. Nous regardons quelle quantité de phéromones se trouvent sur la case adjacente puis nous effectuons un tirage aléatoire selon la répartition des phéromones (c'est que plus une case contient des phéromones, et plus on a de chance de se déplacer sur cette case). Pour que les globules puissent se déplacer dans n'importe quelle direction, il est important d'avoir une quantité de phéromones par défaut (Si, il y a 0 phéromone, la probabilité d'utiliser cette case sera de zéro). Remarquez que le tirage aléatoire est très important, c'est lui qui permet aux globules de trouver de nouvelles voies.

Nous n'avons appliqué la communication par les phéromones que pour le choix du déplacement. Cette méthode pourrait être étendue à tous les choix du jeu. La règle la plus importante à

respecter est que l'on ne doit jamais aller dans une voie avec 100% de chances.

### 3.2.5 Optimisation

La façon la plus triviale de faire faire la mise à jour de phéromones est de parcourir toute la carte et toutes les unités afin de faire évoluer chacune des phéromone. Néanmoins nous stockons aussi des phéromones que l'on pourra dire "inactives", c'est-à-dire qui ont une énergie nulle. Pour ne pas perdre trop de temps on va alors garder à jour une liste des phéromones actives. C'est le rôle de `PheromonePool`.

`PheromonePool` est mit à jour à chaque fois qu'une phéromone passe d'une énergie nulle à une énergie positive. Inversement quand l'énergie passe de positive à nulle la phéromone est supprimée de la liste. L'appel de la fonction `evolue()` se fera donc une seule et unique fois sur le `PheromonePool`.

### 3.2.6 Affichage

Nous avons ajouté quelques fonctionnalités à l'interface pour que le joueur puisse voir l'évolution des phéromones :

- Dans les détails des informations d'un globule on peut voir les informations sur les phéromones. La couleur varie de vert a rouge selon l'énergie.
- Sur la carte s'affichent des sprites représentant les phéromones présents sur celle ci. La taille du sprite varie selon l'intensité.

Toutes les images nécessaires à l'affichage sont chargées lors du chargement du jeu (à l'affichage de la barre de progression).



FIG. 3 – Capture d'écran.

## 4 Listing des modifications apportées

- les globules et le terrain peuvent avoir des phéromones ;
- 4 types de phéromones : ROAD, DANGER, DEAD, SEXUAL ;
- quand un globule passe sur une case il récupère les phéromones de la case et inversement (avec une certaine perte) ;
- les phéromones évoluent chaque tour ;
- ajout au sol et sur le globule des phéromones ROAD (passage) quand un globule sait où il va et qu'il ne se balade pas au hasard ;
- ajout au sol et sur le globule de phéromones DANGER quand un globule ou un bâtiment se fait attaquer ;
- ajout au sol de phéromones DEAD quand un globule ou un bâtiment meurt.
- sauvegarde et chargement de cartes avec phéromones : version 37 du jeu.
- affichage des informations sur les phéromones d'une unité dans l'interface ;
- affichage des phéromones sur la carte ;
- ajout de la décision de déplacement en fonction des phéromones pour les Ouvriers et les Guerriers ;
- ajout de la décision de déplacement en fonction des phéromones pour les Explorers.
- code commenté en Doxygen.
- code marqué par des balises PHEROMONES.

## 5 Bilan

### 5.1 Difficultés rencontrées

Le principal problème rencontré fut la compréhension du code. Effectivement, l'architecture de Globulation est peu appréhendable pour une personne étrangère au code. En particulier, la taille abhérante des fichiers (4000 lignes en moyenne) et des fonctions (les fonctions de 300 lignes sont courantes) ne facilitent pas la compréhension de l'architecture. De plus, le code souffre d'incohérences dans la gestion de certaines fonctionnalités. Le rôle de chaque objet n'est pas clairement défini et on est souvent surpris de voir que l'objet délègue quelquefois une opération et d'autres fois ne le fait pas. Cela entraîne de grosses redondances dans le code et dans les fonctions. Tous ces aspects ne facilitent pas la compréhension de Globulation. De plus, le code manque cruellement de commentaires. des commentaires décrivant précisément le rôle de chaque objet seraient les bienvenus.

Le projet pourrait être grandement amélioré en uniformisant le langage utilisé. Effectivement, bien que le code soit inclut dans des objets, il s'agit de code C. On pourrait utiliser les fonctionnalités du C++ et obtenir un code beaucoup plus compact et lisible. Ensuite, il faudrait réduire la taille des objets à une dimension manipulable, puis, définir le rôle de chaque objet de manière suffisamment précise.

### 5.2 Ajouts possibles

Nous avons pour sujet développer une nouvelle manière de communiquer entre les différents agents. Le code implémenté pourrait être considérablement étendu. Actuellement, il n'intervient que pour l'aide dans le choix d'une direction à prendre lorsque l'agent n'a pas une action précise à faire. C'est à dire que le code implémenté intervient très peu au cours du jeu. Néanmoins, cette méthode de communication pourrait servir de base pour certains algorithmes.

Au cours de l'implémentation de notre sujet, nous avons remarqué la complexité des algorithmes mis en oeuvre pour la gestion des déplacements. Actuellement, les déplacements sont gérés au cas par cas et les phéromones n'intervienne que pour la prise de décision de moindre importance. On pourrait gérer l'intégralité des déplacement à l'aide des phéromones. L'intérêt serait alors d'avoir un code très générique et très adaptable aux nouvelles situations. Ceci permettrait réduire considérablement le code consacré aux déplacements (environ 6000 lignes).