Java Performance - TP5

December 26, 2018

Thomas COUCHOUD
thomas.couchoud@etu.univ-tours.fr
Victor COLEAU
victor.coleau@etu.univ-tours.fr

Chapter 1

Introduction

Le but de ce TP est de mobiliser les différents éléments que nous avons pu voir lors des derniers TPs afin d'améliorer les performances d'une application complète.

Afin d'avoir une meilleure idée de la progression qui a été faite, nous avons découpé notre travail en plusieurs branches qui apportent chacune un différent set d'amélioration.

De plus le fichier ants_worst est celui qui a été utilisé tout le long des tests.

Chapter 2

Branche: victor

Cette branche est la branche de base qui contient un code non modifié. Les tests qui sont effectués afin de calculer le nombre de calcul de déplacements de fourmis par seconde (nombre en bas à gauche de la fenêtre, appelés DPS par la suite) ont tous été réalisés sur la même machine avec Java 8.

Ainsi on observe un DPS de base de l'ordre de 7500 à 8750.

Chapter 3

Branche: tp5.1

Commençons par le premier pack de changements que nous avons effectués. Ces derniers se trouvent sur la branche tp5.1.

Changements effectués:

- Changement des types Objet en types primitifs afin d'éviter un grand nombre de boxing/unboxing.
- Utilisation de List<?> au lieu de Vector<?> (évite la synchronisation).
- Initialisation des matrices de convolution qu'une seule fois de manière statique au lieu de le faire à chaque itération.
- Retirer les mutex inutiles qui génère de la synchronisation (et donc des temps d'attente) pour rien.

Avec ces différents changements nous arrivons à passer à 8500 - 9500 DPS. L'amélioration est présente même si elle est relativement faible. Cependant pour n'avoir changé que des types primitifs et objets utilisés cela reste une amélioration remarquable.

Dans les branches suivantes nous allons tenter d'attaquer le problème plus en profondeur en changeant un peu la partie "algorithmique" du logiciel.

Chapter 4

Branche: tp5.2

Cette nouvelle branche, tp5.2, est basée sur tp5.1. Nous allons donc hériter des changements qui ont déjà été faits.

Changements effectués:

- Utilisation d'un cache d'objet Color.

Le cache vise à éviter la création d'objets Color sans cesse. En effet Color est un objet non mutable et donc nous en créons un nouveau à chaque pixel qui change. De plus ces couleurs sont définies de manière précise et dans un nombre limité. Il est donc possible de réutiliser un objet si la couleur est déjà instanciée.

Afin de réaliser cela, nous avons fait une classe qui reçoit la demande d'instanciation d'une couleur. Si cette dernière existe déjà dans un cache, on renvoie l'instance; sinon on la créée et l'ajoutons à notre cache.

Grâce à ce cache, nous arrivons à un DPS de 9000. Cela n'a pas amélioré les performances comparé à la branche tp5.1 mais a tout de même stabilisé un peu les DPS. Cela peut peut-être être du au fait que le GC gère de lui même le fait que nous réutilisons les objets Color, bien que cela nous semble tout de même étrange qu'il gère ça.

Chapter 5

Branche tp5.3

La branche tp5.3 est basée sur tp5.1. Elle hérite donc des premières modifications mais pas du cache des couleurs.

En effet cette modification ne s'étant pas portée fructueuse, nous avons décidé d'avoir une autre approche en ne stockant que des int pour les couleurs. Cela pourrait éviter de créer pleins d'objets et se rabattre sur des objets primitifs.

Nous stockons donc nos couleurs sous la forme 0xAARRGGBB. Où AA est la transparence, RR le niveau de rouge, GG le niveau de vert et BB le niveau de bleu.

Cependant nous avons eu un petit problème et toutes couleurs sur le canvas sont les mêmes (jaune délavé).

Malgré cela nous restons toujours sur un DPS de 9000 ce qui indique qu'il n'y a pas eu de grande amélioration.

Chapter 6

Branche tp5.4

Cette branche est basée sur tp5.3 et hérite donc de l'utilisation des couleurs sous forme d'int.

Changements effectués:

- Matrices de convolution sous forme d'int, et division à la fin (méthode de convolution).
- Utilisation d'une buffered image pour stocker les couleurs et rafraichissement de canvas directement avec.

L'utilisation de la buffered image nous permet de stocker directement notre tableau de couleurs dans un tableau utilisable par le canvas. En effet auparavant nous utilisions un tableau 2D d'int qui était copié dans le canvas. Grâce à la buffered image on renseigne notre couleur dedans (en s'affranchissant de créer un objet Color pour le pixel à peindre) et écrivons directement notre image sur le canvas.

De plus il est possible de définir l'image à peindre sans avoir à toujours peindre cette dernière dans le repaint. De ce fait notre méthode repaint est maintenant vide et tout se passe dans update. La problématique engendrée par cela est que si l'on pause le rafraichissement du canvas, lors de la reprise nous verrons les fourmis se téléporter (même si l'image a bien été peinte entre temps). Il faudra attendre qu'elles passent dans une zone assez proche pour rafraichir les zones non à jour.

Grâce à cette amélioration nous atteignons les 16000 DPS, ce qu'il est une amélioration plus que conséquence car elle nous permet de double ou presque les DPS de base.

Chapter 7

Branche tp5.5

Branche non terminée qui visait à fixer le problème cité précédemment (zones ne se mettant pas à jour lors de la reprise).