**HE-ARC**

2015-2016

**Rapport et test**

**Laboratoire 1 Mandelbrot**

**Gygi Damien et Jeanneret Cyril**

Table des matières

[Introduction 2](#_Toc439919152)

[Cahier des charges 2](#_Toc439919153)

[Contraintes 2](#_Toc439919154)

[Environnement 3](#_Toc439919155)

[Le programme doit etre fait en langage C sur le framework Code Blocks. pour l’affichage de l’ensemble de Mandelbrot, nous utiliserons la librairie Simple DirectMedia Layer, souvent abrégée SDL. 3](#_Toc439919156)

[Rendu et Barème 3](#_Toc439919157)

[Structure de l’application 3](#_Toc439919158)

[Guide d’utilisation 3](#_Toc439919159)

[Principe de fonctionnement 4](#_Toc439919160)

[Différente partie de code 5](#_Toc439919161)

[Mise en place des threads 5](#_Toc439919162)

[Problèmes rencontrés et solutions 5](#_Toc439919163)

[Résultat obtenu 6](#_Toc439919164)

[Test effectués 6](#_Toc439919165)

[Conclusion 6](#_Toc439919166)

# Introduction

Ceci est le premier laboratoire de programmation concurrente. Nous devons créer un programme pour calculer une courbe de Mandelbrot puis de l’afficher sur l’écran à l’aide de la librairie SDL. Le but est de paralléliser les calculs afin de tirer profit des architectures multi-cœurs.

La courbe de Mandelbrot est construite en appliquant itérativement la fonction Zn+1 =Z2n + C, où C’est le point à évaluer, et Z0 = 0. Le calcul se fait dans le plan complexe, et si une des itérations génère un point en dehors du cercle de rayon 2 centré en (0; 0), alors le point est en dehors de la courbe.

## Cahier des charges

Le but de ce travail pratique est d'implémenter un programme multithread calculant les fractales de l'ensemble de Mandelbrot. Pour ce projet, nous vous fournissons le code d'une implémentation séquentielle qui calcule l'ensemble de Mandelbrot et l'affiche dans une fenêtre graphique à l'aide de la librairie graphique SDL (Simple Direct Media Layer).

Il faut créer un thread master qui s'occupe de diviser le plan complexe à calculer en M blocs ; le master crée ensuite N threads Worker, puis attend qu'ils terminent leur travail.

Les N threads Worker s'occupent chacun de calculer et afficher un des blocs du plan complexe ; lorsqu'un thread Worker a terminé son bloc, il passe alors à un bloc suivant jusqu'à ce qu'il n'y ait plus de bloc à calculer.

**Il est impératif que N soit plus petit ou égal à M ; de plus N et M > 0**

## Contraintes

Le thread main s'occupe de créer le thread master.

• Le programme doit accepter deux arguments sur la ligne de commande : le premier argument est le nombre de blocs et le deuxième le nombre de threads Worker.

• Si l'utilisateur ne spécifie pas exactement deux arguments, alors le programme doit afficher la syntaxe du programme et se terminer.

• le programme doit calculer et afficher le temps mis pour calculer l'image complète.

• il faut s’assurer de désallouer toute mémoire dynamiquement allouée.

• Vous êtes libre d'utiliser les primitives d'exclusion mutuelle ou de synchronisation de votre choix (parmi celles vues jusqu'ici).

• il faut tester le programme avec un nombre varié de valeurs pour M et N ; notez ensuite les timings obtenus.

Essayez de maximiser l'utilisation des processeurs disponibles et évitez toute attente active inutile.

## 

## Environnement

Le programme doit être fait en langage C sur le Framework Code Blocks. Pour l’affichage de l’ensemble de Mandelbrot, nous utiliserons la librairie Simple Direct Media Layer, souvent abrégée SDL.

Il faudra respecter l’architecture « Boss – Worker ». Un thread Master, instancié par le Main, va créer les threads Worker qui calculeront et afficheront l’image. Le nombre de threads qui seront créés par le Master est déterminé par l’utilisateur.

## Rendu et Barème

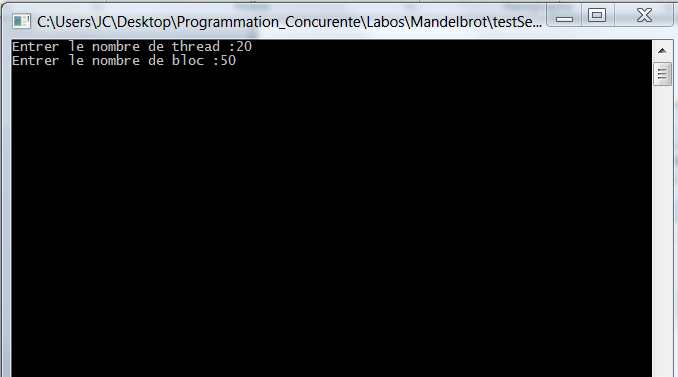
Vous rendez un exécutable avec les dll nécessaires pour son fonctionnement, vous rendez également le code source commenté et un rapport dont vous commentez les tests que vous avez réalisés.

|  |  |
| --- | --- |
| Conception | 10% |
| Exécution et fonctionnement | 30% |
| Tests | 10% |
| Codage | 30% |
| Documentation et commentaires | 20% |

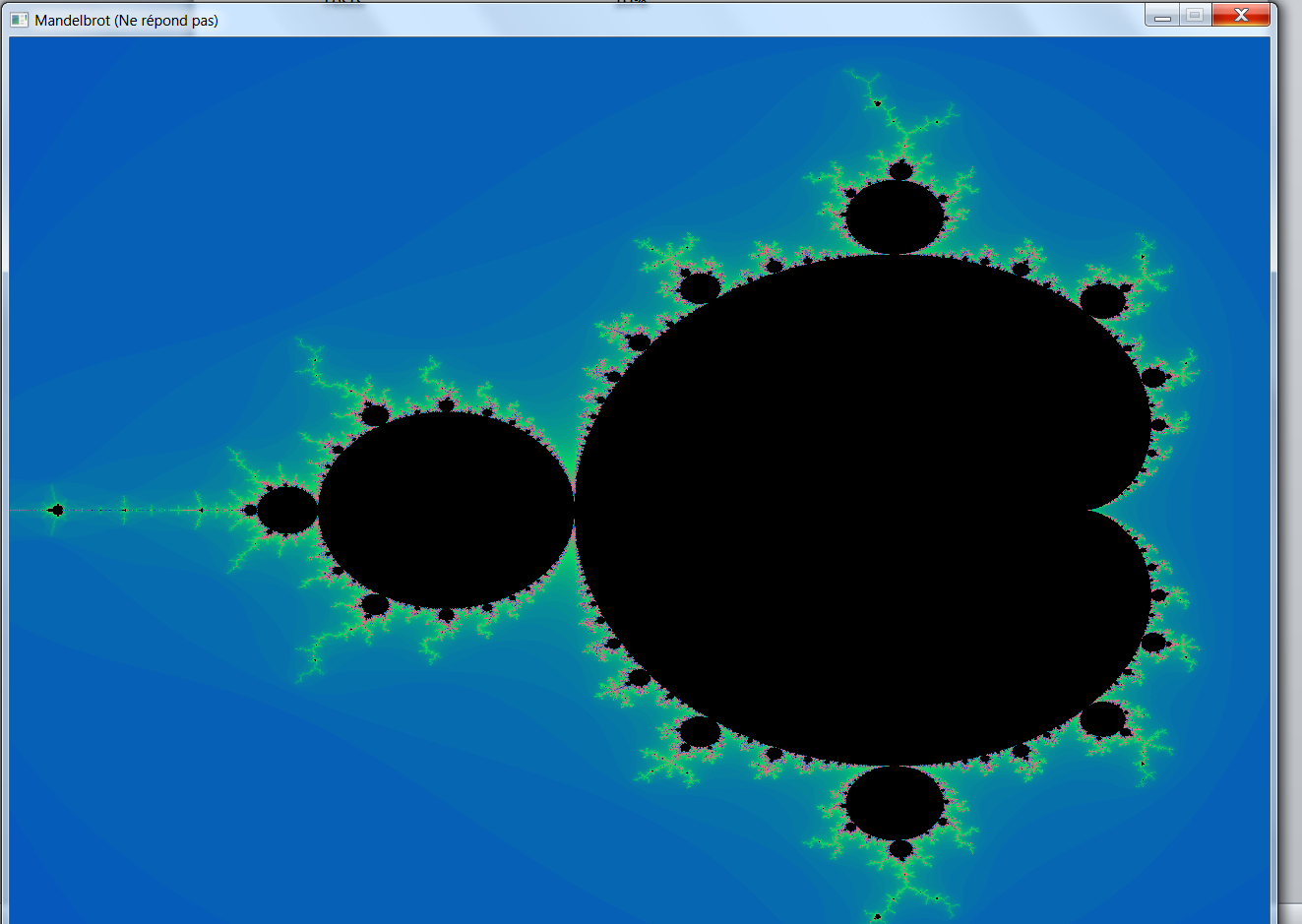
# Structure de l’application

## Guide d’utilisation

Pour utiliser notre application, il suffit d’exécuter le « Lanceur.exe ». Celui-ci va exécuter une console dans laquelle on vous demandera de rentrer le nombre de thread et le nombre de block que vous voulez :



Ainsi ces paramètres vont être envoyés au programme principal qui se chargera d’exécuter les calculs de la Mandelbrot avec les threads. Voici le résultat obtenus :



## Principe de fonctionnement

Notre programme va générer un thread master qui va recevoir tous les paramètres pour le calcul de la Mandelbrot ainsi que les paramètres que l’utilisateur aura choisi pour le nombre de thread et le nombre de blocs. Ce thread master va créer un certain nombre de Thread Worker qui eux vont calculer des colonnes de l’écran et afficher la Mandelbrot. Exemple, notre taille d’écrans en largeur est de 1280 pixel. Si l’utilisateur choisi de séparer cette largeur en 300 blocks et d’utiliser 20 threads, alors ces 5 threads vont se partager ces différentes zone. Voici une démonstration de l’application de ce procéder.

## Différente partie de code

## Mise en place des threads

Pour la partie thread, nous avons créé une structure qui contiendra les différentes informations pour les calculs ainsi que les paramètres de la Mandelbrot. Cette structure est passée en paramètres au thread master. Celui-ci va configurer le nombre de thread Worker dont il a besoins.

## Problèmes rencontrés et solutions

Dans cette partie nous aborderons les problèmes rencontrés ainsi que les solutions ou les moyens de contournement que nous avons trouvé.

# Résultat obtenu

Sdfasdfadf

## Test effectués

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nb Threads | Nb Blocks | 1 exécution  (Temps) | 2 exécutions  (Temps) | 3 exécutions  (Temps) | 4 exécutions  (Temps) | Temps moyen |
| 1 | 1 |  |  |  |  |  |
| 8 | 8 |  |  |  |  |  |
| N multiple 256 (32) | 256 |  |  |  |  |  |

# Conclusion