Ensemble de julia

Projet préparé par :

SAIBI MARZOUK

OULKADI SALIM

PRESENTATION DE PROJET

1-C’est quoi l’ensemble de Julia ?

Les ensembles de Julia ont été découvert par Gaston Julia. Ce sont des fractales du plan complexe obtenus en étudiant les itérations d’applications complexes. Elles referment une explication mathématique bien plus profonde que l’apparence matérielle. Elles s’appuient sur la théorie des variables complexes. On les retrouve dans divers domaines, tels que la physique et
la médecine.

2- Fonctionnement de julia:

Les fractales s’appuient sur les nombres réels et les nombres complexes en analysant leurs suites.
Soit $Zn+1 = Zn^2 + C$ (C est une constante). L'ensemble des valeurs de Z où l'orbite ne tend pas vers l'infini est colorié d'une certaine façon selon le temps d'échappement vers l'infini, les autres en noir.
Les paramètres de couleurs sont en rvb et peuvent être variés de 0 à 255, pour obtenir une couleur différente. La forme de la fractale peut être changée en variant la valeur de la constante complexe C. Pour cela, on entre deux valeurs : partie réelle et partie imaginaire.

3-Côté programmation :

Pour obtenir une fractale de Julia en c/c++ nous avons eu recourt à la librairie OpenCv comme indiqué dans les instructions.
Étant curieux, intéressés et motivés, nous avons fait nos propres recherches afin de bien comprendre le principe qui se cache derrière les belles images que renvoie la Fractale de Julia ;
nous n’avons donc pas repris le code donné Nous n’avons donc pas
Le principe est le suivant :

//Avant le Main

1- Faire les différents includes.
2- Création des constants WIDTH et HEIGHT de la fractale de Julia.
3- fonctions qui nous serviront à parcourir l’image (pixel par pixel).
4- Création et association d’un pointeur à une image.
5- Création de deux fonctions zoom (height et width).
6- Création de la fonction Julia.

7- Création de deux Thread.(le premier pour zoomer et l'autre pour dézoomer

//Dans le Main

1- Déclaration du deux Thread.

2- Création d'une image 800*600.

3- Création d'une fenêtre pour y charger notre image.

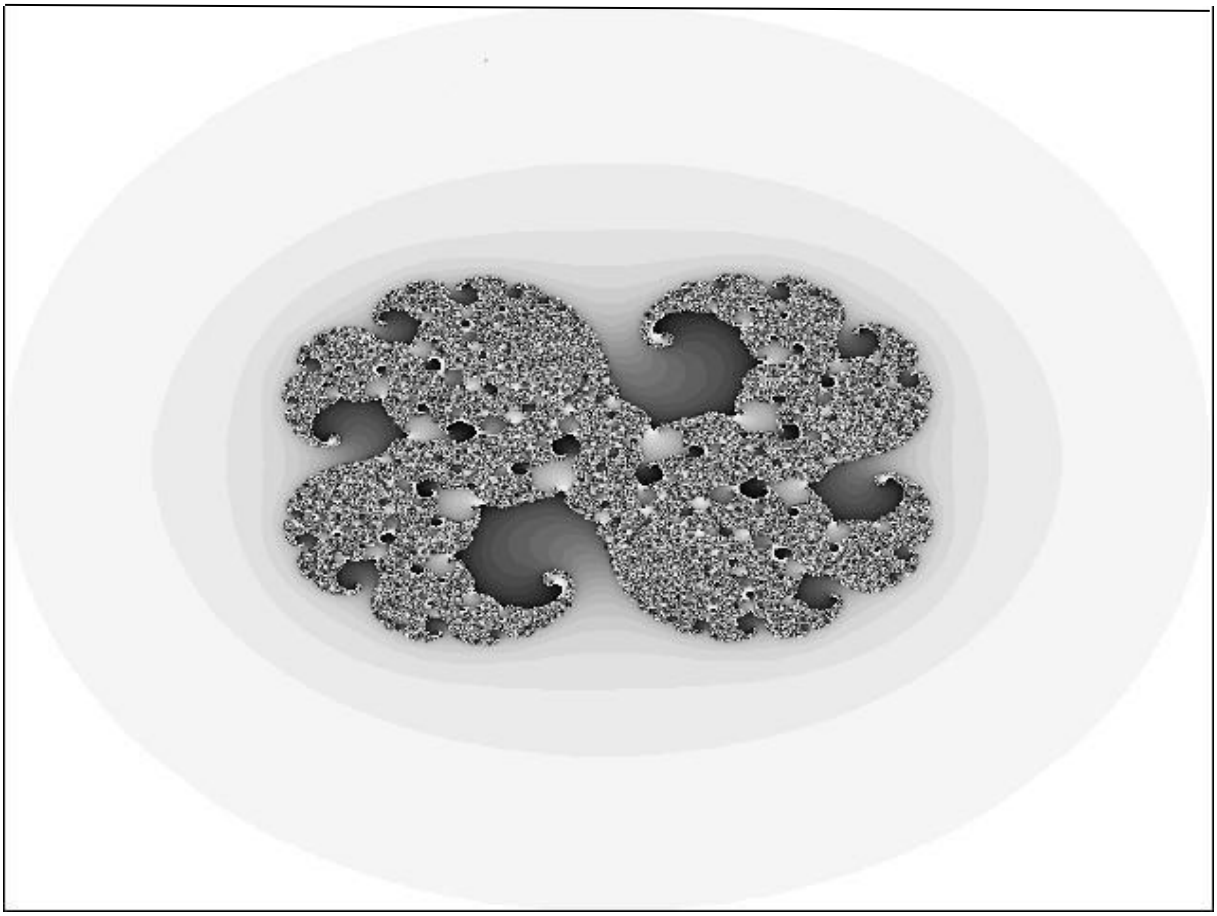
4- Appel au thread-1 pour zoomer en cliquant sur le
bouton « c »

5-appel au thread-2 pour dézoomer en cliquant sur le
bouton « x » lorsque le zoome est terminer

6- Affichage de l'image (chargement dans la fenêtre).

4-Exemple de fractale :

Voici un petit exemple de fractale obtenu avec un nombre réel **re = 0,013** et un nombre
imaginaire **im = 0,285.**



5- Quelques options (Interactivité):

-Le zoom :

Le zoom est géré de façon automatique par le 6ème paramètre de la fonction Julia. Une fonction gx() et gy() préalablement crées agrandissent la fractale à l'aide d'un simple calcule mathématique intégrant la variable **zoom.**

-Itération :

Les itérations sont faites de façon automatique afin d'avoir une sorte d'animation. Pour ce faire, nous réitérons automatiquement les calcules.

-Break :

Un simple appuie sur la touche Q du clavier suffit pour quitter l'animation.

6- Intégration des Threads:

En programmation temps réel, les threads permettent de faire plusieurs tâches en parallèle.

Parmi leurs nombreux intérêts, on pourrait citer l'optimisation des programme du fait de la diminution du temps d'exécution, comme dans notre cas, avec la fractale de Julia par exemple.

Pour ce faire, le principe est de diviser notre fractale en plusieurs sections qui seront alors prises en charge par des threads, chacune indépendamment de l'autre.

Le but est que chaque section soit prise en charge par un thread, les calculs seront moins grands et donc le temps d'exécution le sera également.

***Important :**

A noter que les threads sont à doubles tranchons ; si on en ajoute trop, ça ralentira le temps d'exécution, il est donc capital de choisir le bon nombre de threads, le nombre optimum pour que le temps d'exécution de notre fractale soit optimal.

7- Comment compiler le programme :

Pour compiler le programme il nous suffit d'ouvrir le terminal dans le dossier le contenant puis saisir ces deux lignes :

g++ -g -Wall -o view julia.cpp `pkg-config opencv --libs --cflags` -lpthread

./view

*Analyses & critiques du graphe :

On distingue trois phases principales dans notre graphes :
Au départ, on voit que même en augmentant le nombre de threads, le temps de d'exécution de la fractale n'est pas influencé. Le temps gagné en exécution de la fractale est perdu lors du chargement des threads.
Arrivé à un certain point, le temps d'exécution devient inversement proportionnel au nombre de threads ; plus le nombre de threads augmente, plus le temps d'exécution diminue, jusqu'à avoir le plus bas temps d'exécution de la fractale. Le nombre de threads varie en fonction de la fonction Julia utilisée ainsi que des performances de la machine.
Le point optimum dépassé, le temps d'exécution devient proportionnel au nombre de threads utilisés; il augmente tout en augmentant le nombre de threads.
Concrètement, on atteint la limite des performances du processeur de la machine, il ne peut prendre en charge autant de threads, leurs temps de chargement augmentent et l'image met donc du temps à s'afficher du fait qu'elle est divisée en petites sections qui sont générées par les threads.