Machines Parallèles

Projet 1 : zoom sur une fractale

Rapport

Énonce

« Le but de ce projet est de réaliser une vidéo montrant un zoom sur un point particulier d'une fractale »

- On pourra utiliser **Opencv** pour générer la vidéo
- On pourra utiliser une structure de donnée image **multirésolutions** où la résolution est d'autant plus précise que l'on est à proximité du point d'intérêt

Organisation du code :

Ce projet comporte deux fichiers : ***julia.cpp*** et le ***Makefile***

On commence par importer les bibliothèques nécessaires telles que **complex** afin d'établir la formule qui créé notre fractale, **opencv2/opencv.hpp** ainsi qu'**opencv2/videoio.hpp** pour générer la vidéo.

Vec3b julia_pix(float x, float y) ;

Cette fonction retourne une couleur qui permet de différencier s'il on est dans la fractale ou non, respectivement jaune ou rouge.

void julia(Mat out, float zoom, int begin, int end) ;

C'est dans cette procédure, que nous implémentons le zoom dans les coordonnées **x** et **y**

int main() ;

Dans le main, on déclare une matrice **Mat image(H, W, CV_8UC3);** qu'on envoie à **julia** avec la variable **zoom** de type **float** initialisée à **4.00**, puis, celle-ci sera décrémentée afin de pouvoir zoomer sur la fractale.

On initialise la vidéo qu'on va créer à l'aide de **VideoWriter**, chaque séquence d'images qu'on affichera sera inscrite dans la video grâce à video.**write(image);** définie plus bas dans le **while**.

Le **while** :

Si la valeur de la variable **zoom** est négative, on sort de la boucle pour éviter de dézoomer.

On affiche ensuite avec **imshow** chaque séquence d'images avant de créer les **8 thread** qu'on join a chaque tour de boucle**.**

La variable **key** récupère la valeur retournée par **waitKey()** pour ensuite être comparée avec le nombre **27** et **113** correspondant respectivement aux codes de la touche **Echap** et le caractère « **Q** ». Si la condition est vraie alors on fait un **break** afin de sortir du **while**.

Enfin, on appelle **destroyAllWindows();** pour détruire la fenêtre ouverte.

Processus de parallélisme :

On créer dans le main, **8 thread**, qui se verront chacun attribuer une partie de l'image, pour cela, on augmente le nombre de paramètres de la fonction **julia** en lui envoyant en plus, le début ainsi que la fin de la coordonnées **r** sur laquelle, le thread doit travailler.

On adapte également la formule, des coordonnés **(x, y)** de julia, à la matrice **out**.

- float y = -(float(r) - **out.rows** / 2) * zoom / **out.rows**;
- float x = (float(c) - **out.cols** / 2) * zoom / **out.cols**;

Comparaison des Temps d'exécution :

Avec **0 thread** :

```
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ g++ -W -Wall -std=c++17 -O3
 test.cpp `pkg-config --cflags --libs opencv` -pthread
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ time ./a.out

real    3m27,659s
user    3m30,052s
sys     0m0,489s
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$
```

Avec **2 thread** :

```
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ g++ -W -Wall -std=c++17 -O3
 test.cpp `pkg-config --cflags --libs opencv` -pthread
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ time ./a.out

real    0m56,163s
user    1m51,295s
sys     0m0,364s
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$
```

Avec **4 thread** :

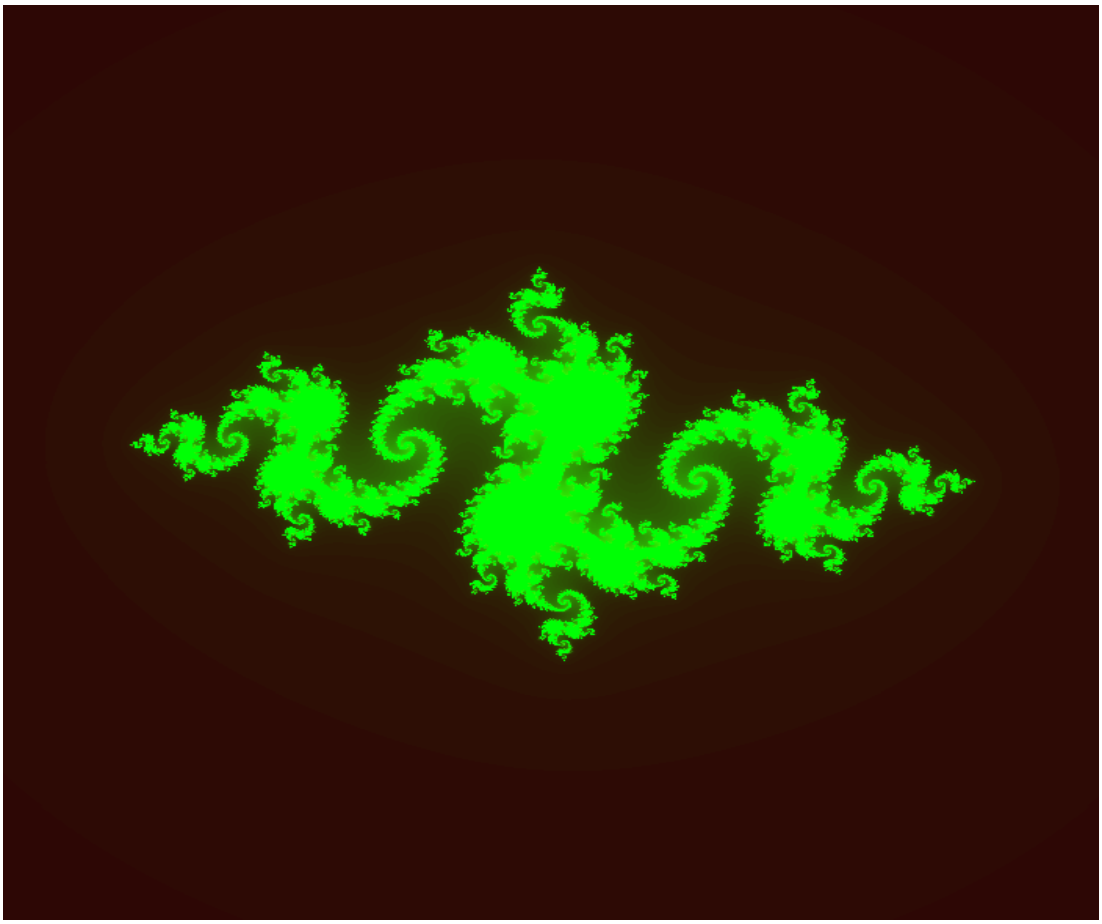
```
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ time ./a.out

real    0m20,061s
user    1m0,099s
sys     0m0,212s
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$
```

Avec **6 thread** :

```
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ g++ -W -Wall -std=c++17 -O3
 test.cpp `pkg-config --cflags --libs opencv` -pthread
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ time ./a.out

real    0m14,132s
user    0m39,532s
sys     0m0,192s
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$
```

Avec **8 thread** :

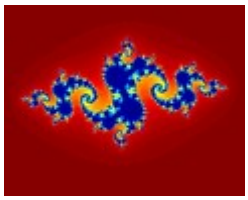
```
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ make
g++ -W -Wall -std=c++17 -O3 -pthread julia.cpp `pkg-config --cflags --libs opencv`
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$ time ./a.out

real    0m9,210s
user    0m31,053s
sys     0m0,152s
diana@diana-ThinkPad-T430:~/Documents/julia-with-opencv$
```



Problèmes non résolus :

- Lorsqu'on zoom sur la fractale, celle-ci n'est pas régénérée, donc pas de zoom sur la fractale indéfiniment.
- Il n'y a pas de structure de donnée image **multirésolutions.**

Bibliographie :



Ensemble de Julia rempli pour $c = -0{,}8+0{,}156\,i$.

http://www.ai.univ-paris8.fr/~jnv/mpl/

https://fr.wikipedia.org/wiki/Ensemble_de_Julia

https://opencv.org/