Samuel CIULLA

PROGRAMMATION PROCÉDURALE

TRAITEMENT D'IMAGE

DOSSIER n° 2

Introduction

Le projet comporte la réalisation d'un programme informatique en langage C donnant la possibilité de créer des images à partir d'images principales que l'on modifie.

Le programme a été créé avec l'interface de développement CLion de la suite JetBrains sur un MacOS 10.13.

La dernière version du programme rendue et qui satisfait tous les critères requis est la version 2 (la 3ème).

Les fonctions

Le programme permet de traiter des images de plusieurs façons:

Le pavage simple

Cette technique consiste à diviser l'image en pavés égaux et pour chaque pavé d'extraire la valeur de chaque canal (R, G et B) de chaque pixel présent dans le pavé et d'en calculer la valeur/couleur moyenne, enfin de colorier tout le pavé avec cette couleur. La fonction utilisée à cet effet est:

```
void simple_paving(image *img, int factor);
```

- *img*: pointeur vers la structure d'image source;
- *factor*: la taille d'un côté d'un pavé

Tests effectués

Le pavage simple a été testé en faisant varier le paramètre *factor*. La fonction a été écrite pour la gestion des entiers uniquement. Vu que l'image de source est carrée de préférence et que les pavés doivent tous être égaux, la gestion des flottants s'est révélée superflue. Exemple de test: ***simple_paving(img, 20);***

Le mosaïque

Cette technique se base sur le *pavage simple* mais traite chaque pavé différemment. Dans chaque pavé est ajoutée la miniature de l'image source (ou une autre image de même rapport de dimensions) et la couleur moyenne est calculée en tenant compte de cet ajout. Pour que cette fonctionnalité soit possible il est nécessaire de créer une autre fonctionnalité qui permette de changer la taille d'une image donnée, la rétrécir dans ce cas:

```
void resize_img(image *img, int factor);
```

- *img*: pointeur vers la structure d'image source;
- *factor*: facteur d'échelle pour le redimensionnement

```
void mosaic(image *img, char *rs_img, int factor);
```

- *img*: pointeur vers la structure d'image source;
- *rs_img:* nom de l'image choisie qui sera placée dans chaque pavé
- *factor*: facteur d'échelle pour le redimensionnement

Tests effectués

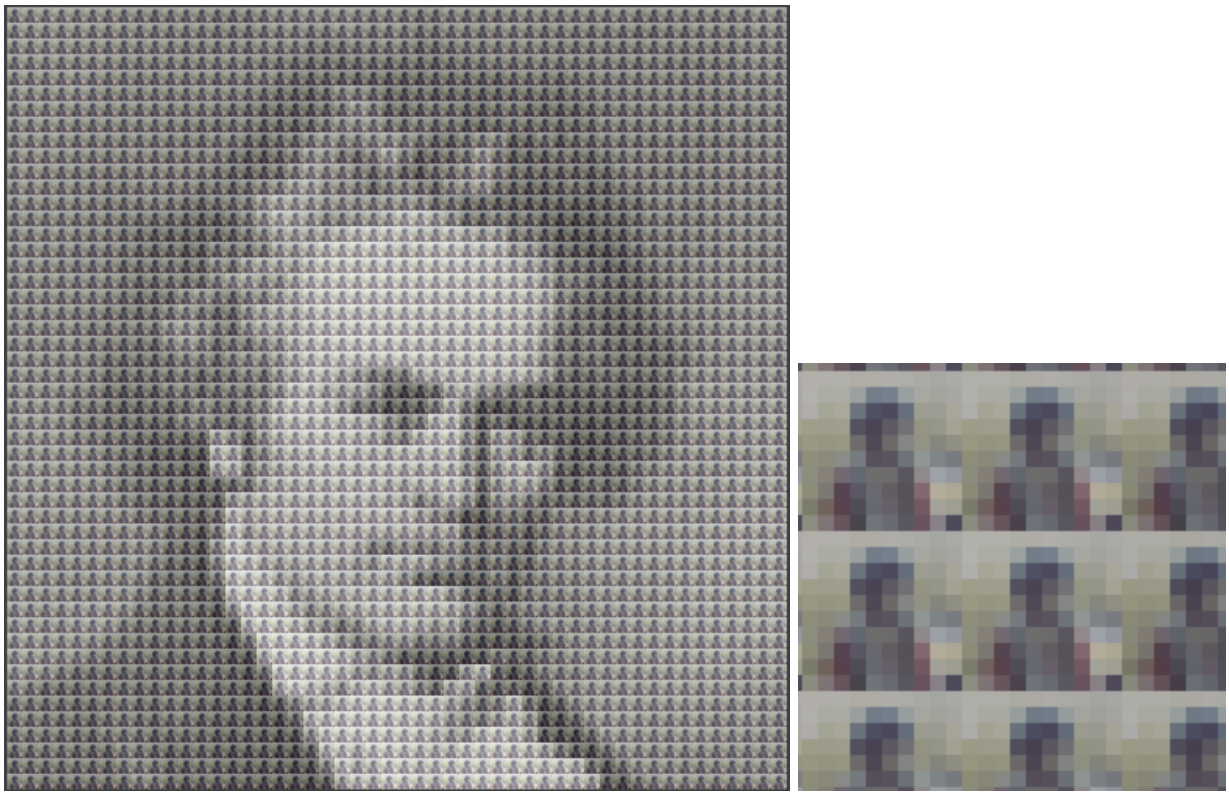
De même que pour le *pavage simple,* la fonction *resize_img* a été testée en faisant varier le paramètre *factor* afin de redimensionner l'image donnée en tailles différentes. Cette fonction, ainsi que la fonction *mosaic* ont été écrites pour gérer les entiers uniquement et les valeurs choisies pour le paramètre factor ont été choisies à cet effet. Pour la fonction *mosaic*, l'image de miniature a également été modifiée durant les tests.

Exemple de test: ***resize_img(img, 10);***

500x500 BMP (24-bit color) 750,14 kB

50x50 BMP (24-bit color) 7,65 kB

Exemple de test: ***img = Lire_Image("beethoven", ""); mosaic(img, "hendrix", 25);***



Le donut

Cette fonctionnalité permet de convertir les coordonnées cartésiennes de l'image de source en coordonnées polaires afin de représenter l'image étirée en forme circulaire. Pour que ceci soit possible il faut d'abord calculer la taille de l'image en sortie et allouer l'espace mémoire nécessaire. Une fonction à part a été prévue à cet effet (*set_background*) qui, en plus du calcul de la taille et de l'allocation de l'espace mémoire, s'occupe aussi d'initialiser l'arrière-plan avec une couleur unie, qui se trouve être la couleur moyenne de l'image de source:

```
image * polarize(image *in, char *in_name, double angle, uint radius, int cut_dir);
```

- *in*: pointeur vers la structure d'image source;
- *in_name*: nom du fichier de l'image de source;
- *angle*: définit l'angle du donut, si l'angle est 360 le donut sera entier;
- *radius*: définit si le donut aura un trou au centre et permet d'en paramétrer la taille;
- *cut_dir*: définit l'orientation de l'image.

```
image * set_background(image *img, char *img_name, int radius);
```

- *img*: pointeur vers la structure d'image source;
- *img_name:* nom du fichier de l'image de source;
- *radius*: servant à calculer la taille exacte de l'image de sortie

Tests effectués

Les tests effectués sur la fonction *polarize* ont porté sur la variation des paramètres *angle*, *radius* et *cut_dir*. La fonction *set_background* est appelée par la fonction *polarize* donc son test se fait automatiquement.

Exemple de test: ***out = polarize(img, “landscape”, 270, 100, S);***