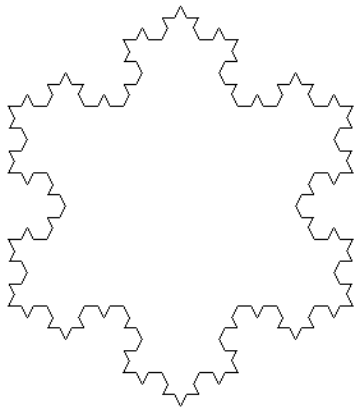
Fractales de Koch

Ensimag 1A - Préparation au Projet C



Présentation

Le but de cet exercice est de se familiariser avec quelques points de base du langage C, à travers la mise en œuvre de fractales de Koch.

Les objectifs de ce sujet, du point de vue du langage C, sont les suivants :

- Programmation modulaire
- Allocation dynamique
- Utilisation des arguments argc et argv de la fonction main
- Utilisation du préprocesseur C
- Ecriture dans un fichier
- Utilisation des types C99
- Manipulation des opérateurs binaires <<, >>, |, &
- Gestion des erreurs
- Utilisation des fonctions de la librairie string (strcmp, …)
- Utilisation du debugger gdb/ddd et de valgrind

Nous vous demandons d'utiliser des types C99 pour vos variables : uint32_t, uint16_t, int32_t, int16_t, bool, etc…

Le debugger **gdb/ddd** sera utilisé pour tracer les erreurs du programme. **valgrind** sera aussi utilisé pour vérifier l'allocation correcte des données dynamiques du programme.

Ce TP fera l'objet d'un rendu individuel sur TEIDE, sous la forme d'une archive contenant l'ensemble des fichiers source (.c, .h) nécessaires à la compilation de votre projet, ainsi qu'un Makefile (celui fourni par les enseignants, ou le vôtre si vous l'avez modifié).

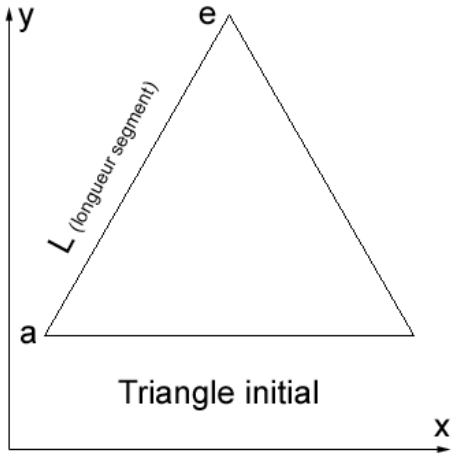
1 Programme : Fractale géométrique Flocon de Koch

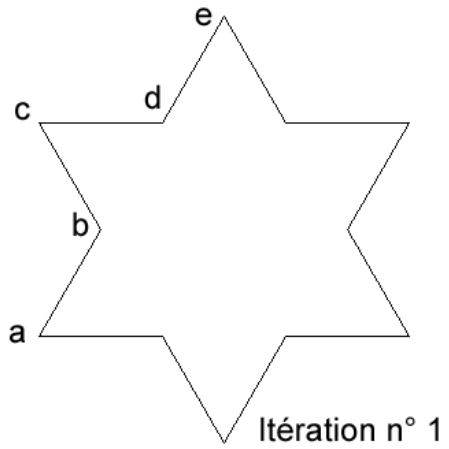
2.2 Principe

Il s'agit de la première fractale inventée en 1904 par le mathématicien suédois Helge von Koch (1870 – 1924).
Pour tracer cette courbe, il faut:

- Tracer un triangle équilatéral
- Remplacer le tiers central de chaque côté par une pointe dont la longueur de chaque côté est égale aussi au tiers du côté
- Recommencer cette construction sur chaque côté des triangles ainsi formés.

Schéma des étapes successives :

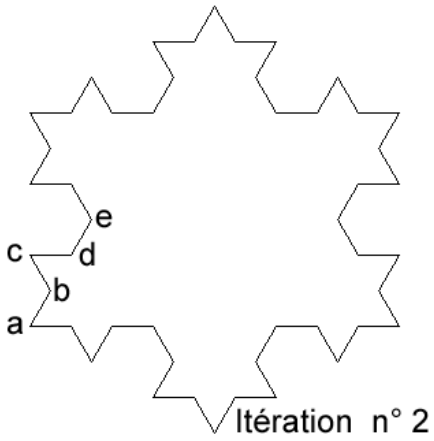




Pour chaque segment [a,e], on peut calculer les coordonnées des points intermédiaires b,c et d de la façon suivante :

```
xb = xa + (xe-xa)/3
yb = ya + (ye-ya)/3
xd = xa + 2 * (xe-xa)/3
yd = ya + 2 * (ye-ya)/3
xc = (xb+xd) * cos(60°) - (yd-yb) * sin(60°)
yc = (yb+yd) * cos(60°) + (xd-xb) * sin(60°)
```

A l'itération suivante, on recommence sur les nouveaux segments créés.



2.2 Objectifs

Le programme Flocon de Koch devra permettre de paramétrer les données suivantes :

- La longueur (en pixels) d'un segment du triangle initial.
- La taille en pixels (hauteur = largeur) du carré de l'image finale sera calculée automatiquement à partir de la longueur de segment précédente.
- Le nombre d'itérations à effectuer
- La couleur de tracé (qui pourra être codée sous la forme hexadécimale 0xRRVVBB, RR : rouge, VV : vert, BB : bleu)
- La couleur de fond (qui pourra être aussi codée sous la forme 0xRRVVBB)
- Le nom du fichier .ppm de destination
- Une dernière option « all » indiquant qu'on veut tous les fichiers image intermédiaires entre le triangle de départ et le flocon de koch final. Si elle est omise, seul le fichier image final est généré.

Ces paramètres principaux de la fractale seront stockés dans une seule structure de données.

Les paramètres pourront être donnés en arguments de l'appel du programme en ligne de commandes :

- Exemple de ligne de commande : koch 270 4 0xFF0000 0xFFFFFF koch.ppm all
- Cet exemple lancera le programme en définissant la taille d'un segment du triangle initial à 270 pixels, le nombre d'itérations de calcul à 4, 5 fichiers de sortie .ppm (00_koch.ppm, 01_koch.ppm, 02_koch.ppm, 03_koch.ppm, 04_koch.ppm,), les images seront rendues avec un tracé rouge sur fond blanc

Si aucun paramètre n'est passé en arguments de la ligne de commande ou si le nombre d'arguments attendus est incomplet, le programme Koch demandera à l'utilisateur de saisir les paramètres en mode interactif.

2.3 Directives de programmation

2.3.1 Programmation modulaire

Le programme Koch comportera au minimum 4 modules

- Module principal (programme principal) : koch_main.c
- Module fonctions (les fonctions liées au traitement des données du programme : calcul, rendu,...) : koch_fonctions.c
- Module ihm (l'interface homme machine contenant les fonctions liées à l'affichage et la saisie des données) : koch_ihm.c
- Module image (création de l'image au format ppm) : create_image.c

Chaque module sera découpé en fonctions élémentaires permettant une programmation simple, structurée et très lisible du programme.

2.3.2 Structure de données

Une liste chainée est imposée pour stocker tous les points calculés du flocon de Koch. Chaque élément de la liste contiendra les coordonnées X et Y d'un point du flocon de Koch et un lien vers le point suivant.
Les coordonnées seront typées en entiers non signés uint32_t.

2.3.3 Etapes de résolution

Il est demandé de procéder de la façon suivante :

- Définition de la liste chaînée initiale des points définissant le triangle de départ ;
- Génération et calcul des points du flocon de Koch. Les nouveaux points seront insérés dynamiquement à la liste chainée initiale ;
- Rendu image par la méthode Bresenham (détaillée ci-dessous). Cette méthode permettra de tracer des traits entre les points calculés du flocon de Koch. L'image générée sera stockée dans un tableau de type uint32_t * alloué dynamiquement en mémoire ;

- Ecriture de l'image mémoire uint32_t * dans le fichier .ppm de destination à l'aide d'une fonction create_image_ppm.

2.3.4 Méthode de tracé de ligne Bresenham

Documentation sur la méthode : http://en.wikipedia.org/wiki/Bresenham's_line_algorithm
On propose de mettre en œuvre la méthode générale simplifiée en se basant sur le pseudo code suivant :

```
/* Tracé de ligne entre 2 points de coordonnées (x0,y0) et (x1,y1) */
function line(x0, y0, x1, y1)
    dx := abs(x1-x0)
    dy := abs(y1-y0)
    if x0 < x1 then sx := 1 else sx := -1
    if y0 < y1 then sy := 1 else sy := -1
    err := dx-dy

    loop
        setPixel(x0,y0)
        if x0 =  x1 and y0 = y1 exit loop
        e2 := 2 * err
        if e2 > -dy then
            err :=err – dy
            x0 := x0 + sx
        end if
        if e2 < dx then
            err := err + dx
            y0 := y0 +sy
        end if
    end loop
```

2.3.5 Représentation interne de l'image

Il est demandé pour des raisons pédagogiques de représenter l'image générée comme un tableau d'entiers non signés de 32 bits. Une image sera donc représentée dans votre projet par une variable de type *uint32_t **.
Chaque élément de ce tableau est composé de 4 octets, dont 3 serviront à stocker la valeur d'une composante couleur (R, V ou B) d'un pixel. Le dernier octet sera inutilisé.

La représentation suggérée considère qu'une variable de type *uint32_t* contient dans ses trois octets de poids faible trois valeurs représentant des intensités de couleurs dans l'ordre R, V, B (du poids le plus fort vers le poids le plus faible).

Exemple :

```
uint32_t * image;
...
/* Exemple d'un élément/pixel du tableau image initialisé avec une couleur codée en hexadécimal */
image[...] = 0x00F788AA;
/* Composante Rouge de la couleur du pixel : 0xF7 soit 247 en décimal */
/* Composante Verte de la couleur du pixel : 0x88 soit 136 en décimal */
/* Composante Bleue de la couleur du pixel : 0xAA soit 170 en décimal */
...
```

Vous pourrez écrire des primitives permettant de lire et d'écrire les composantes de couleur R, V et B dans un entier uint32_t, en vous appuyant sur les opérateurs binaires du langage C.

2.3.6 Création d'une image de type ppm

Pour pouvoir dessiner la fractale de Koch et l'afficher à l'écran, le programme principal permettra de générer un fichier au format d'image PPM en sortie. Les fonctions *fopen*, *fwrite* (ou *fputc*) et *fclose* devront être utilisées pour la création de ce fichier de sortie.

2.3.6.1 Création du fichier PPM

Le format PPM (Portable Pixel Map) consiste en un en-tête spécifiant le type d'image (ex : couleur ou noir et blanc) et ses dimensions, puis la suite des pixels de l'image ligne par ligne, chaque pixel étant codé sur 3 octets : un pour le rouge (R), un pour le vert (V) et un pour le bleu (B).
Il est possible d'aller voir à l'adresse http://netpbm.sourceforge.net/doc/ppm.html ou simplement le « man ppm » pour plus de détails sur ce format. Néanmoins, dans le cadre du travail demandé, le fichier devra commencer par l'en-tête suivant :

```
P6
LARGEUR HAUTEUR
255
```
où LARGEUR et HAUTEUR sont la largeur et la hauteur de l'image ***au format texte*** *et 255 la valeur maximale de chaque couleur (rouge, vert ou bleu)*

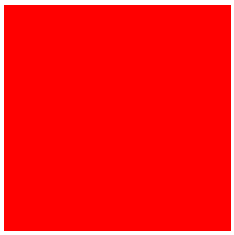
Notez bien que, comme l'indique la description du format P6, l'entête du fichier doit être écrit au format texte, alors que **la suite des pixels de l'image devra être écrite au format binaire dans le fichier**.

Implémenter une fonction create_image_ppm ayant comme paramètres la variable image uint32_t *image, la largeur et la hauteur de l'image codées en int32_t et le nom du fichier de sortie codé sous forme de chaîne de caractères.

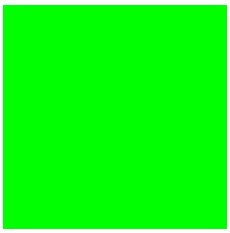
2.3.6.2 Test et visualisation du fichier PPM

Le format d'images PPM est visualisable par beaucoup de visionneurs, comme par exemple **eog** (eye of gnome) sous linux/Unix. Si vous souhaitez convertir vos fichiers PPM en images JPG, vous pouvez utiliser la commande **ppmtojpg**, avec l'option **--smooth=30** pour lisser le résultat.

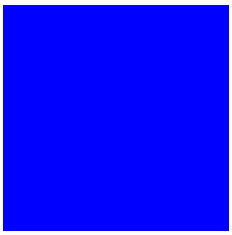
Ce module peut en effet être testé indépendamment du reste du projet : vous pouvez par exemple construire des structures de données représentant des images particulières « à la main » (par exemple : image carrée de couleur, dessin de diagonale) dans un fichier de test et vérifier que la génération de fichier PPM par un appel à **create_image_ppm** correspond bien au résultat attendu.
Par exemple, la génération des images ci-dessous permet de vérifier la bonne prise en compte de la couleur :



rouge.ppm

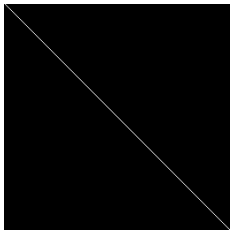


vert.ppm



bleu.ppm

Celle-ci teste l'orientation de l'image :



diagonale.ppm

2.3.7 Code initial fourni

Afin d'orienter la réalisation du code demandé, les fichiers entêtes .h des modules fonctions et ihm sont proposés :

- koch_fonctions.h
- koch_ihm.h
- create_image.h

Ils contiennent en particulier :

- Les définitions des structures de données pour la liste chaînée et les paramètres du programme
- Les prototypes proposés pour les fonctions principales à mettre en oeuvre

3 Extensions possibles (en option)

- Figure de départ différente d'un triangle (figure à n segments : carré, hexagone, ...).
- Variantes des courbes de Koch (http://fr.wikipedia.org/wiki/Flocon_de_Koch) :
  - Fractales Cesàro
  - Courbes de Koch quadratiques