TP 7 : PGCD et Fractales

Notions : Récursivité

1 Calcul du PGCD

Le calcul du pgcd de 2 nombres a et b peut se faire de manière simple, en utilisant la récursivité selon l'algorithme suivant : pour a supérieur ou égal à b, le pgcd de a et b est :

1. a, si b est égal à 0,
2. le pgcd de b et du reste de la division de a par b, sinon.

2 Un arbre fractal

Il existe des courbes du plan qui présentent des auto-similarités et sont infiniment imbriquées dans elles-mêmes : certaines parties sont semblables à la courbe complète. Ces sont des fractales. Elles sont de longueur infinie, bien que contenues dans une surface d'aire finie.

Quelques unes de ces fractales sont définies par un processus récursif très simple, à partir d'un segment initial auquel on applique récursivement des transformations linéaires (rotation, homothétie, translation).

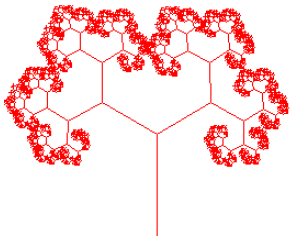

FIGURE 1 – Un arbre fractal.

Par exemple, la courbe de l'arbre (figure 1) est définie de manière récursive par le processus suivant :

1. Pour commencer, on prend un segment AB (en noir sur la figure ci dessous),
2. on crée le segment BM_1 par une homothétie de valeur r, suivie d'une rotation de θ degrés autour du centre A puis d'une translation $\overrightarrow{OB}$ du segment initial,
3. on crée le segment BM_2 par une homothétie de valeur r, suivie d'une rotation de $-\theta$ degrés autour du centre A puis d'une translation $\overrightarrow{OB}$ du segment initial,
4. on répète, jusqu'à atteindre l'ordre désiré, les étapes 2 et 3 sur les nouveaux segments en rouge sur les exemples :
   (a) BM_1 d'une part,
   (b) BM_2 d'autre part.

On modélise ces transformations par 3 fonctions : f_1 pour le segment BM_1, f_2 pour le segment BM_2 et f_3 pour le segment AB. Ce processus s'appelle IFS, pour Iterated Function System. En prenant les points $A(0,0)$ et $B(x,y)$, les fonctions s'écrivent :

$$f_1(x,y) = r.\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}.\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

$$f_2(x,y) = r.\begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}.\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

$$f_3(x,y) \quad = \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

La figure 2 représente les trois premiers ordres de la courbe de l'arbre pour $r = \sqrt{2}$ et $\theta = \pi/4$.

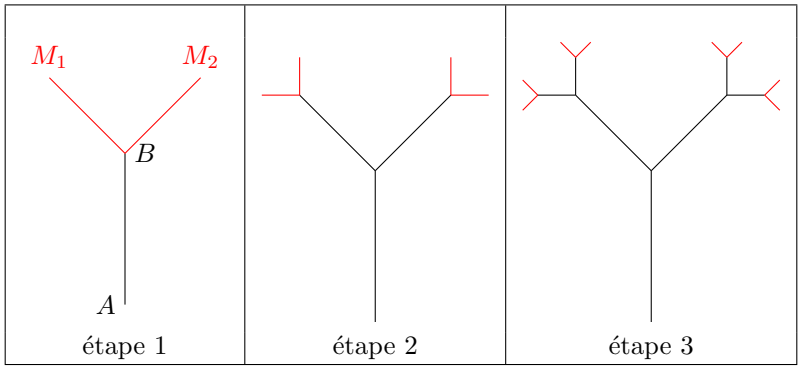

FIGURE 2 – Les 3 premiers ordres de la courbe.

Si les points initiaux sont $A(x_a, y_a)$ et $B(x_b, y_b)$, la fonction $f_1(x,y)$ donne le point $M_1(x_1, y_1)$, la fonction $f_2(x,y)$ donne le point $M_2(x_2, y_2)$. On conserve le segment AB grâce à la fonction $f_3(x,y)$.

Les calculs des coordonnées des points $M_1(x_1, y_1)$ et $M_2(x_2, y_2)$ s'écrivent :

$$\begin{array}{rcl} x_1 & = & x_b + r * \cos(\theta) * (x_b - x_a) - r * \sin(\theta) * (y_b - y_a) \\ x_2 & = & x_b + r * \cos(\theta) * (x_b - x_a) + r * \sin(\theta) * (y_b - y_a) \\ y_1 & = & y_b + r * \sin(\theta) * (x_b - x_a) + r * \cos(\theta) * (y_b - y_a) \\ y_2 & = & y_b - r * \sin(\theta) * (x_b - x_a) + r * \cos(\theta) * (y_b - y_a) \end{array}$$

Pour dessiner l'arbre fractal, il faut donc utiliser une procédure récursive qui prend en paramètres les points entre lesquels il faut tracer la courbe, ainsi que le niveau (ou ordre) que l'on veut atteindre. Il faut tracer la droite entre les deux points. Si le niveau est atteint, on s'arrête. Sinon, on crée les deux nouveaux segments et on relance la procédure récursive de dessin sur chacun de ces segments.

---

tracer la droite entre les deux points A et B
if le niveau n'est pas atteint **then**
  calculer les coordonnées de M_1 et de M_2
  incrémenter le niveau
  relancer la procédure récursive sur chacun des nouveaux segments BM_1 et BM_2
else
  ne rien faire !!
end if

2.1 Version en mode texte

1. Ecrivez une fonction qui affiche les coordonnées des points M_1 et M_2 calculés à chaque niveau, sans utiliser de fonction graphique.
2. Ecrivez un programme qui appelle cette fonction.

Les valeurs des premiers points M sont les suivantes pour $A(100, 0)$ et $B(100, 100)$ pour $r = 2/(1+\sqrt{5})$ et $\theta = \pi/4$ en arrondissant les coordonnées :

— étape 1 : (56,143) et (143,143)

— étape 2 : (17,142) (55,181) et (143,180) (180,143)

3 Programmation multi fichier et makefile

Vous allez maintenant séparer votre code en plusieurs fichiers et réaliser un fichier `Makefile` qui sera utilisé par la commande `make` pour compiler. Voici les fichiers à construire, dans un nouveau répertoire, à partir du programme que vous venez d'écrire :

1. un fichier d'entête `fractale.h`, qui contient les prototypes des fonctions et les définitions des pseudo constantes de type `#define M 6`. Il est identique à celui que vous venez de faire.
2. un fichier source `fractale.c`, qui contient le code des fonctions. Il est identique à celui que vous venez de faire, si ce n'est qu'il ne contient pas de fonction principale `main`.
3. un fichier source `programme.c`, qui contient le code de la fonction `main`, identique à votre fonction précédente. Ce fichier doit inclure le fichier d'entête `fractale.h`, mais aucun code autre que `main`.
4. un fichier `Makefile`, dans lequel vous allez définir les règles utiles à la compilation de vos deux fichiers sources pour construire un exécutable :
   (a) définir une variable CFLAGS pour les options de compilation `CFLAGS=-c -g -Wall`
   (b) définir une variable LDFLAGS pour les options d'édition des liens `LDFLAGS=-lm`
   (c) définir la règle de construction de votre exécutable, baptisé `programme`

   ```
   programme : fractale.o programme.o
    clang $(LDFLAGS) fractale.o programme.o -o programme
   ```

   (d) définir la règle de compilation du fichier source `fractale.c`

   ```
   fractale.o : fractale.c
    clang $(CFLAGS) fractale.c
   ```

   (e) définir la règle de compilation du fichier source `programme.c`

   ```
   programme.o : programme.c
    clang $(CFLAGS) programme.c
   ```

Sauver votre fichier `Makefile` puis compiler avec la commande `make`

4 Version graphique

4.1 Bibliothèque phelma

Vous devez utiliser la bibliothèque graphique SDL pour réaliser l'affichage graphique. Vous pouvez créer une fenêtre graphique dans laquelle votre fonction récursive va dessiner quand cela sera utile. Cette fenêtre, créée dans le programme principal, sera passée comme paramètre à la fonction récursive pour que celle-ci puisse dessiner dedans.

Trois étapes sont nécessaires pour dessiner dans une fenêtre graphique :

1. déclaration d'une variable de type `SDL_PHWindow*`
2. création d'une fenêtre vide par la fonction `SDL_PH_CreateWindow`.

   Cette fonction prend en paramètres deux entiers `nbc` et `nbl`, qui indiquent respectivement le nombre de colonnes (largeur) et de lignes (hauteur) de la fenêtre à créer. Elle retourne un pointeur vers la fenêtre créée ou `NULL` si cette ouverture est impossible.
3. dessin d'un trait dans la fenêtre graphique : `lineColor`

   Cette fonction prend 6 paramètres. On l'appelle de la façon suivante :

   ```
   lineColor(f->rendu, xdep, ydep, xarr, yarr, color);
   ```

   Elle trace une droite de couleur `color` dans la fenêtre `f->rendu`, entre les points `(xdep,ydep)` et `(xarr,yarr)`.

Vous retrouvez ces trois étapes dans le programme contenu dans le fichier `exemple.c`. Ce dernier réalise les actions suivantes :

1. il crée une fenêtre vide,
2. il trace une ligne noire entre les points de coordonnées $(x = 10, y = 0)$ et $(x = 300, y = 50)$,
3. il met l'écran à jour,
4. il attend un retour chariot pour se terminer.

FIGURE 3 – Dessin d'un trait dans la fenêtre graphique (l'origine (0,0) se trouve en haut à gauche).

Pour compiler, il faut copier le fichier `Makefile` (voir le site des tp) et utiliser la commande unix `make exemple` pour créer un exécutable. On lance l'exécution par la commande `./exemple`.

```
#include <SDL2/SDL_phelma.h>

int main (int ac, char **av) {
  /* Variable permettant de manipuler une fenetre */
  SDL_PHWindow* f1=NULL ;

  /* Creation d'une fenetre de dimension 300x100,
     Couleurs sur 32 bits, Fond de couleur blanche
     CRTL-C possible pour tuer le programme */
  f1=SDL_PH_CreateWindow(300,100);

  /* Verification de l'ouverture de la fenetre ;
        sinon, arret du programme */
  if (f1==NULL) {
      puts("impossible d'ouvrir une fenetre graphique") ;
      exit(EXIT_FAILURE) ;
  }

  SDL_PH_ClearWindow(f1); /* efface la fenetre graphique */

  /* Affichage : dessiner sur la fenetre f1
     une ligne noire entre les points (x=10,y=0) et (x=300,y=50)
     Attention, la ligne n'est pas encore visible a l'ecran */
  lineColor(f1->rendu, 10, 0, 300, 50, SDL_PH_BLACK);

  /* Met l'ecran a jour : la ligne apparait */
  SDL_PH_FlushWindow(f1);

  /* un getchar pour attendre, sinon le programme se termine
        et la fenetre disparait aussitot */
  puts("Taper une touche pour continuer") ;
  getchar() ;

  /* libere la memoire */
  SDL_PH_DestroyWindow(f1);
  SDL_Quit();
  return EXIT_SUCCESS;
}
```

4.2 Travail à réaliser

1. Quels sont les paramètres de la fonction dont le rôle est de dessiner la courbe précédente à l'ordre n sur une fenêtre à l'écran ?
2. Ecrire une fonction puis un programme dessinant la courbe précédente à l'ordre n, la valeur de n étant saisie au clavier.