Représentation de la fougère de Barnsley

Zakaria TBER
Alexis TREILLES

Mai 2023

Encadrant :
GIRAUD Rémi

1 Introduction

Ce projet vise à construire une image représentant la fougère de Barnsley, une figure fractale géométrique. L'objectif est de créer une représentation visuelle de cette figure en utilisant des paramètres tels que la position de départ, l'orientation et la taille approximative. Le projet se base sur des règles de construction et de récurrence pour générer la figure finale. Une approximation initiale de l'hexagone est proposée pour faciliter le développement et le débogage. Les conditions initiales spécifient les dimensions de l'image finale à créer.

2 Création d'une image :

Nous avons utilisé le format Portable Pixmap (PPM) est associé à l'extension .PPM et permet une programmation rapide des fonctions de lecture et d'écriture. Un fichier PPM contient les éléments suivants dans cet ordre : l'en-tête "P6", un octet de séparation, la largeur de l'image en pixels, un octet de séparation, la hauteur de l'image en pixels, un caractère de séparation, la valeur maximale d'intensité des composantes de couleur (255), un dernier caractère de séparation, et les valeurs des composantes de pixels au format binaire.

2.1 La stucture color :

```
typedef struct color{
    unsigned char red;
    unsigned char green;
    unsigned char blue;
} color;pixels;
} picture;
```

cette structure nous permet de définir une couleur RGB facilement durant tout le projet.

2.2 La structure picture :

```
typedef struct picture{
    int width;
    int height;
    color* pixels;
} picture;
```

Cette structure est utilisée pour siplifier le tracé. Elle comporte la taille de l'image ainsi qu'un pointeur de couleur. Lors de l'écriture du fichier.ppm on est contraint d'écrire les pixels lignes par ligne. Cette structure nous pemet donc de definir l'ordre de nos pixels dans l'ordre qui nous convient.

2.3 Création de la structure picture en fonction de la taille désirée de l'image :

```
picture new_pic(int width , int height){
    picture pic;

    pic.width = width;
    pic.height = height;
    pic.pixels = (color*)malloc((pic.width*pic.height)*sizeof(color));
    color white = {255, 255, 255}; // D finition de la couleur blanche

    for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
            set_pixel(pic, x, y, white);
            // Affecter la couleur blanche    chaque pixel de l'image
        }
    }

    return pic;
}
```

Ici on créée simplement une variable de type picture avec la taille choisis. On en profite par la meme occasion pour parcourir tous les pixels et de definir leur couleur sur blanc pour des raisons esthétiques.

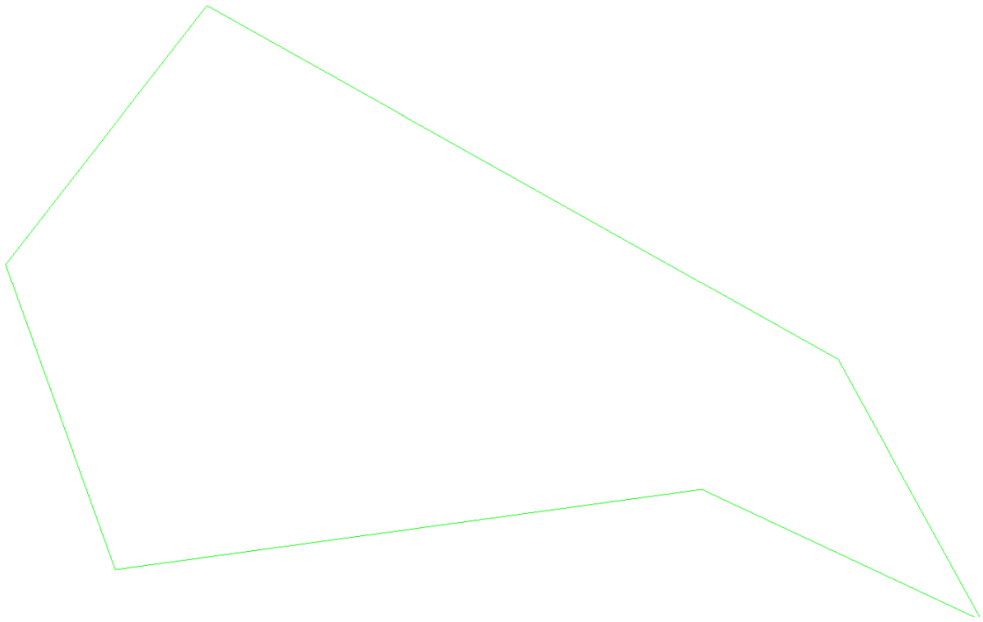
2.4 Sauvegarde de l'image au format .ppm :

```
void save_pic(picture pic, char* image){
    FILE* im;
    im = fopen(image, "wb+");
    fprintf(im, "P6\n%d %d\n255\n", pic.width, pic.height);
    for (int i = 0; i < pic.height*pic.width-1; i++){
        color c = pic.pixels[i];
        fputc(c.red, im);
        fputc(c.green, im);
        fputc(c.blue, im);
    }
    fclose(im);
}
```

Cette fonction parcours tous les pixels de notre structure image et écrit tous les pixels selon la normalisation .ppm exposée ci-dessus.

3 Dessin du polygone :

Dans le sujet on nous propose de tracer la figure suivante pour savoir ou se positionneront les différentes branches de notre fougère.



3.1 pisitionnement d'un pixel :

```
    void set_pixel(picture pic,int x, int y, color col){

    if (x >= 0 && x < pic.width && y >= 0 && y < pic.height) {
            pic.pixels[y*pic.width+x] = col;
        }
}
```

Ici on vérifie que les coordonées sont bien dans l'image et ensuite on assigne la couleur au coordonées correspondante.

3.2 Tracé d'une ligne :

```
void draw_line(picture pic, int x1, int y1, int x2, int y2, color color){
  int i;
  int n=((abs(x1-x2)>=abs(y1-y2))*abs(x1-x2))+((abs(x1-x2)<abs(y1-y2))*abs(y1-y2))
      +1;
  for (i=0;i<n;i++)
      if (n>1)
        set_pixel(pic,(int)(((double)(x2-x1)/(n-1)*i)+x1),(int)(((double)(y2-y1)/(n
            -1)*i)+y1),color);
      else
        set_pixel(pic,(int)x1,(int)y1,color);
}
```

On prend les 2 points qui compose la ligne et on calcule tous les point entre les 2 pour leur assigner une couleur a l'aide de set pixel.

3.3 Tracé de la figure :

afin de simplifier les nombreux tracé qu'implique la récurence on reprend le principe des vecteurs en liste chainé vu lors des TP précédents.
On défini cette chaine de vecteurs de la maniere suivante :

```
typedef struct vector{
    double x1;
    double x2;
    double y1;
    double y2;
    struct vector* next;
}vector;
```

On reprend le principe de lecture des vecteurs dans un fichier texte :

```
vector* read_vector_file(char* filename) {
    vector* first = NULL;
    vector* current = NULL;
    FILE* file = fopen(filename, "r");

    float x1, y1, x2, y2;
    while (fscanf(file, "%f %f %f %f", &x1, &y1, &x2, &y2) == 4) {
        //printf("Vecteur lu\n");
        vector* new_vector = malloc(sizeof(vector));
        new_vector->x1 = x1;
        new_vector->y1 = y1;
        new_vector->x2 = x2;
        new_vector->y2 = y2;
        new_vector->next = NULL;

        if (first == NULL) {
            first = new_vector;
            current = new_vector;
        }
        else {
            current->next = new_vector;
            current = new_vector;
        }
    }
    fclose(file);
    return first;
}
```

On aloue la mémoire aux vecteurs, on les lis un par un et les aloue a une variable de type vecteurs.

3.4 Tracé de la figure adapté :

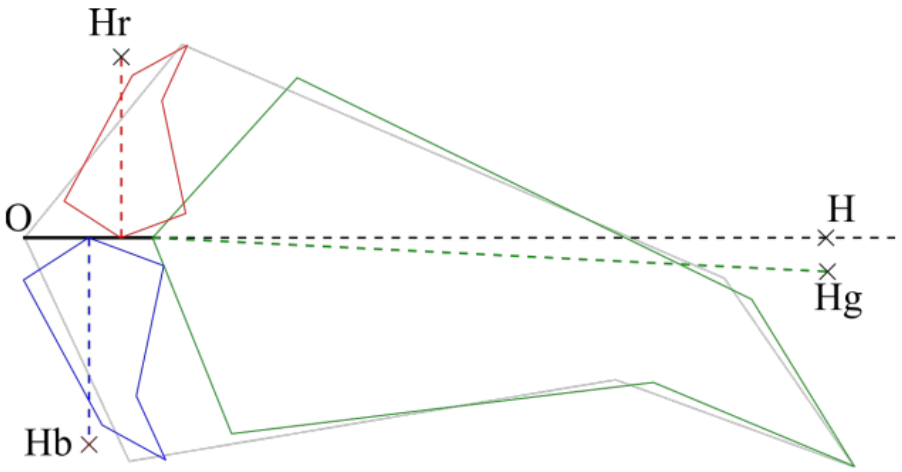
Afin de simplifier le processus lors du tracé de la fougère on créée une fonction qui prend les coordonées du point O de la figure, l'echelle de tracé de la figure ainsi que son angle d'inclinaison.

```
void draw_fig(picture pic, point pt, double scale, double angle, color col, int
    sens) {
        vector* vect2 = read_vector_file("vectorbackup.txt");
        vector* vect = read_vector_file("vectorbackup2.txt");
    if(sens==0){
        vect=vect2;
    }
    while (vect != NULL) {
        //mise a l'echelle des vecteurs ainsi qu'au bon point de d part
        vector* new_vector = malloc(sizeof(vector));
        new_vector->x1 = pt.x + (vect->x1 * scale);
        new_vector->y1 = pt.y + (vect->y1 * scale);
        new_vector->x2 = pt.x + (vect->x2 * scale);
        new_vector->y2 = pt.y + (vect->y2 * scale);
        new_vector->next = NULL;
        // rotation si l'angle est nin nul
        if (angle != 0) {
            double cos_angle = cos(angle);
            double sin_angle = sin(angle);
            double x1 = new_vector->x1 - pt.x;
            double y1 = new_vector->y1 - pt.y;
            double x2 = new_vector->x2 - pt.x;
            double y2 = new_vector->y2 - pt.y;
            new_vector->x1 = pt.x + x1 * cos_angle - y1 * sin_angle;
            new_vector->y1 = pt.y + x1 * sin_angle + y1 * cos_angle;
            new_vector->x2 = pt.x + x2 * cos_angle - y2 * sin_angle;
            new_vector->y2 = pt.y + x2 * sin_angle + y2 * cos_angle;
        }

        // Dessiner le vecteur ( on ne dessine que les points pour une raison
            est tique)
        //draw_vector(new_vector, pic, col);

        //Dessiner les points O,A,B,C,D,E,H
        set_pixel(pic,pt.x,pt.y,col);     // Point H
        set_pixel(pic,new_vector->x1,new_vector->y1,col);  // Point O
        set_pixel(pic,new_vector->x2,new_vector->y2,col);  // Points A,B,C,D,E

        vect = vect->next;
    }
}
```

On a aussi ajouté une variable sens, qui nous servira par la suite a tracer la figure dans le sens shouaité, la figure inversée étant dans "vectorbackup2.txt".

4 Récurence :

4.1 Calcul des différents points pour le tracé des figures suivantes :



On calcule les points tels qu'ils nous sont proposé dans le sujet, on conserve aussi leur noms.

```
      pthr.x = pt.x + 0.12 *sc * cos(ang);
      pthr.y = pt.y + 0.12 *sc* sin(ang);
      pthb.x = pt.x + 0.08 *sc * cos(ang);
      pthb.y = pt.y + 0.08 *sc* sin(ang);
      pt.x = pt.x + 0.16 *sc * cos(ang);
      pt.y = pt.y + 0.16 *sc* sin(ang);
      draw_line(pic, pthr1.x,pthr1.y,pt.x,pt.y,col);
      draw_line(pic, pthb.x,pthb.y,pt.x,pt.y,col);
```

On trace les lignes qui rejoignent les différents points calculé, ce qui nous tracera la tige de notre fougère.

4.2 Détermination du sens de chaque fougère :

```
if (sens==0){
        ang=ang+0.05;
        // trac  de la figure
        draw_fig(pic,pthb,sc*0.27,ang+3.1416/2,col,1);
        draw_fig(pic,pthr,sc*0.3,ang-3.1416/2,col,0);
        if (sc>taille){
            // appel de la r curence si la taille (scale) est superieure a 10
            pic=recurence(  it,  ang+3.1416/2, sc*0.27, pic, pthr,col,1);
            pic=recurence(  it,  ang-3.1416/2, sc*0.27, pic, pthb,col,3);
     }
  }
  //gestion du sens de trac  de la figure et de l'angle de courbure de la foug re
  if (sens==1){
        ang=ang-0.05;
        draw_fig(pic,pthb,sc*0.27,ang+3.1416/2,col,1);
        draw_fig(pic,pthr,sc*0.3,ang-3.1416/2,col,0);
        if (sc>taille){
            pic=recurence(  it,  ang+3.1416/2, sc*0.27, pic, pthr,col,1);
            pic=recurence(  it,  ang-3.1416/2, sc*0.27, pic, pthb,col,3);

     }
  }
  //gestion du sens de trac  de la figure et de l'angle de courbure de la foug re
  if (sens==2){
        ang=ang-0.05;
        draw_fig(pic,pthr,sc*0.27,ang+3.1416/2,col,1);
        draw_fig(pic,pthb,sc*0.3,ang-3.1416/2,col,0);
        if (sc>taille){
            pic=recurence(  it,  ang+3.1416/2, sc*0.27, pic, pthr,col,0);
            pic=recurence(  it,  ang-3.1416/2, sc*0.27, pic, pthb,col,2);

     }
  }
  //gestion du sens de trac  de la figure et de l'angle de courbure de la foug re
  if (sens==3){
        ang=ang+0.05;
        draw_fig(pic,pthb,sc*0.27,ang+3.1416/2,col,1);
        draw_fig(pic,pthr,sc*0.3,ang-3.1416/2,col,0);
        if (sc>taille){
            pic=recurence(  it,  ang+3.1416/2, sc*0.27, pic, pthr,col,1);
            pic=recurence(  it,  ang-3.1416/2, sc*0.27, pic, pthb,col,3);

     }
  }
```

On appele la récurence en fonction de chaque cas on adapte la variable sens afin de tracer la figure dans le bon sens et pour permettre aussi de geré la direction de courbure de la fougère.

5 Résultat final :