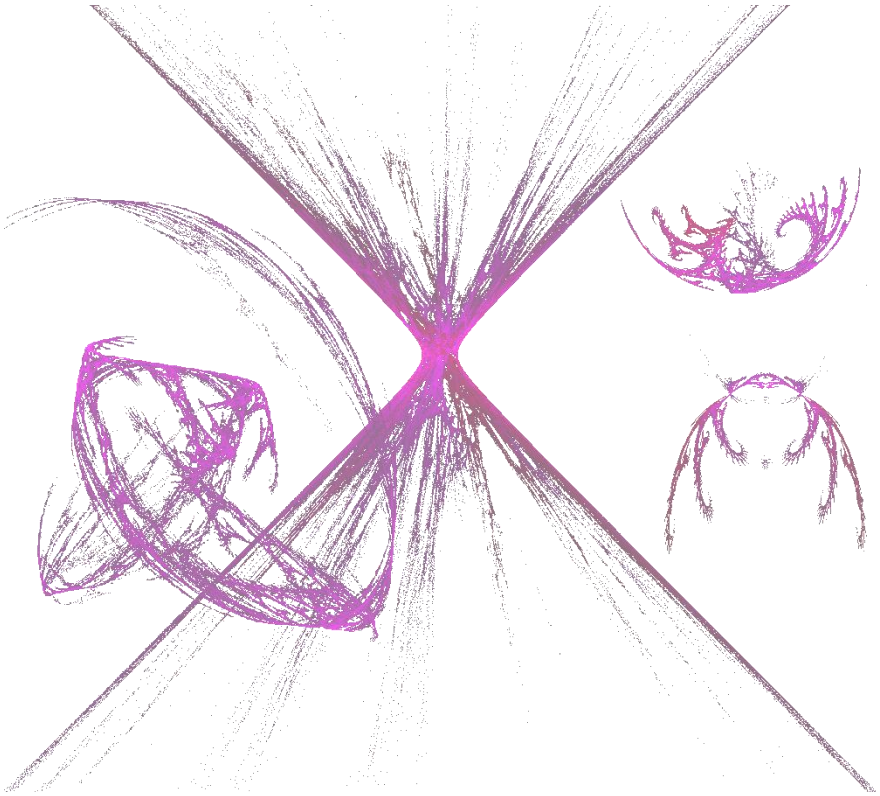
15/01/2012

EISTI

RAPPORT DE PROJET : FRACTALE



LEDIT Alexandre | SOUPLET Antoine

Table des matières

Introduction

Une figure fractale ou une fractale, est une courbe, une surface ou un volume qui est définit selon des règles déterministes ou stochastiques (c'est-à-dire régies par des lois mathématiques bien définies ou par des lois aléatoires) de forme irrégulière. Le mot « Fractale » a été inventé en 1974 par le mathématicien Benoît Mandelbrot à partir de la racine latine « fractus » qui signifie brisé ou irrégulier. Il est considéré comme le père des fractales et une des configurations fractales porte son nom. Les travaux qui lui ont progressivement permis de dégager ce concept débutèrent dans les années 50.

La particularité des Fractales la plus connue est l'invariance par changement d'échelle, en effet dans certains types de fractales, la forme dite « de base » se répète à une échelle plus petite à « l'intérieur » d'elle-même, ce qui créé des répétitions à l'infini. Cette particularité est aussi appelée par les mathématiciens homothétie interne.

Il n'existe pas à proprement parler de définition pour les fractales car il existe de nombreux types différents qui ne répondent pas aux mêmes lois.

Dans ce projet nous avons implémenté cinq de ces différents types de fractales, nous allons détailler le modèle mathématique associé à chacun de ces cinq types.

Ensemble de Mandelbrot

L'ensemble de Mandelbrot est aujourd'hui un des modèles de fractale les plus connus. En 1979, Benoit B. Mandelbrot s'intéressa à une équation de récurrence complexe toute simple :

$$\begin{cases} Z_{n+1} = Z_n^2 + C \\ Z_0 = 0 \end{cases}$$

La valeur de C étant une constante complexe qu'il associa à un point de l'écran de son ordinateur (où la partie réelle correspond à l'abscisse et la partie imaginaire à l'ordonnée). Pour chaque nombre complexe C associé à un pixel de son écran, il obtint une suite de nombres complexes.

Il calcula le module de chacun des termes de la suite. Lorsque la suite des modules ne tendait pas vers l'infini, le point C était considéré comme appartenant à l'espace recherché et était noirci.

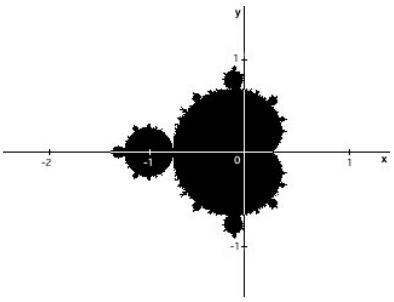


Image 1 : Ensemble de Mandelbrot

Nous avons coloré, comme le montre l'image ci-dessous, les points à l'extérieur de l'ensemble de Mandelbrot en utilisant des couleurs qui dépendent du nombre de termes calculés avant d'obtenir un module supérieur ou égal à 2. Les points d'une même couleur peuvent être interprétés comme étant des points s'éloignant à la même vitesse de l'ensemble de Mandelbrot.

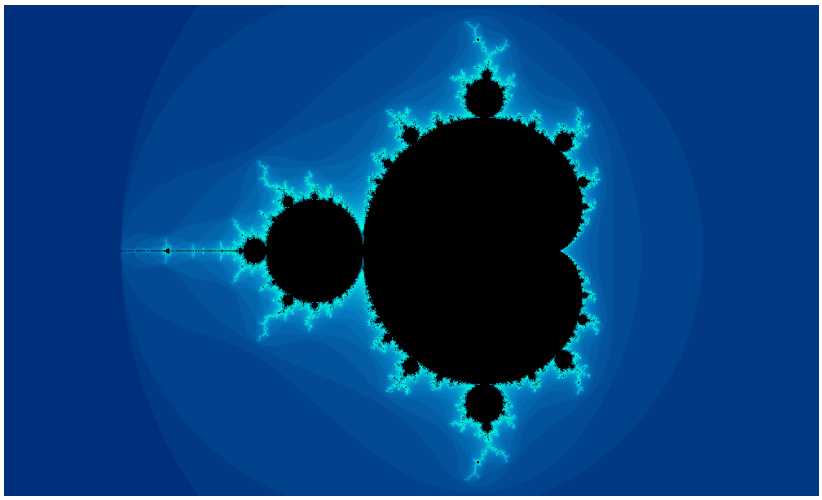


Image 2 : Ensemble de Mandelbrot

Ensemble de Julia

L'ensemble de Julia est définit exactement comme celui de Mandelbrot grâce à la formule de récurrence complexe :

$$Z_{n+1} = Z_n^2 + C$$

Pour obtenir un élément de l'ensemble de Julia, on procède à l'inverse. On fixe une valeur de C et on fait varier Z_0. D'abord on construit une grille, comme on l'a fait pour l'ensemble de Mandelbrot. On choisit un point sur cette grille (à l'intérieur ou à l'extérieur de l'ensemble de Mandelbrot). Ce point correspondra la valeur de C. On donne ensuite à Z_0 chacune des valeurs de la grille. Pour chaque valeur associée à Z_0, on obtient une suite de nombres complexes. Si la suite des modules de ces nombres complexes ne tend pas vers l'infini ($|Z_n| < 2$ pour tout n), on noircit le point associé à Z_0. Autrement, on colorie ce point d'une couleur correspondant au nombre de termes qu'on a dû calculer avant d'obtenir un module supérieur ou égal à 2. Lorsque tous les points de la grille ont été associés à Z_0, on obtient une image colorée. Cette image est une fractale de type Julia.

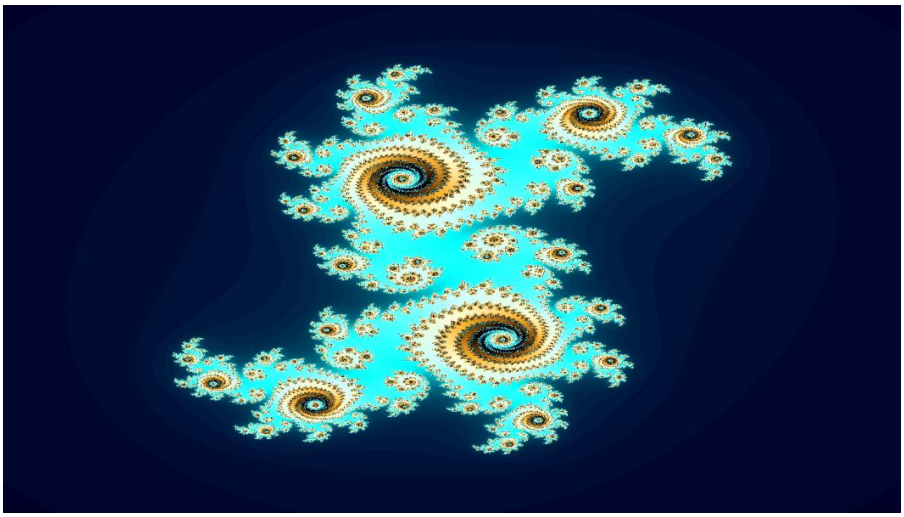


Image 3 : Ensemble de Julia

L-System

Un L-System est un ensemble de règles et de symboles syntaxiques qui pourrait être apparenté à un langage de programmation très basique. Ces règles modélisent des algorithmes de croissance. Une « phrase » est définie récursivement par substitution de caractères avec une nouvelle « phrase ». Cette « phrase » est ensuite analysée caractère par caractère chacun correspondant à une action de géométrique :

Loi : a -> a+a++a

Etat 0 : a+a

Etat 1 : (a+a++a) + (a+a++a)

Etat 2 : ((a+a++a)+ (a+a++a)++ (a+a++a)) + ((a+a++a)+ (a+a++a)++ (a+a++a))

Où :

- a est l'action tracer un segment
- + l'action pivoter d'un angle défini

La répétition de ce procédé fait apparaitre des figures complexes, toutes impliquant des homothéties internes.

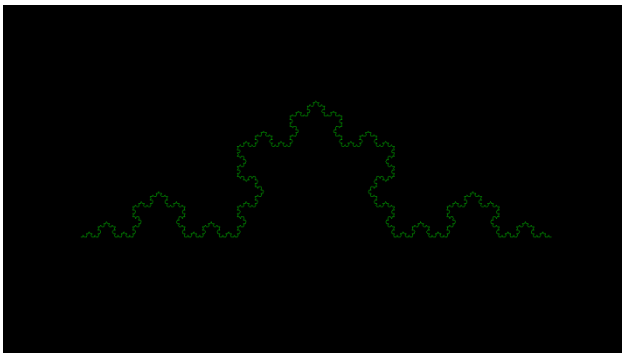


Image 4 : Fractale de type L-System (Koch)

Iterated Function System (IFS)

Un IFS est un système de fonctions mathématiques itérées qui lorsqu'on l'applique à un point permet d'obtenir une fractale.. En d'autres termes, partant d'un point initial auquel on applique une des fonctions du système, choisie aléatoirement on obtient un nouveau point auquel on réapplique le même traitement. Les fonctions du système, sont affines et contractantes. Le terme contractant signifie que lorsque l'on applique une de ces fonctions à un point de façon itérative, on obtient une représentation graphique contenue dans un espace fini (les coordonnées de ce point peuvent toujours être majorées).

Les fonctions qui sont itérées sont sélectionnées aléatoirement parmi ce système de fonctions, toutes définies de la manière suivante :

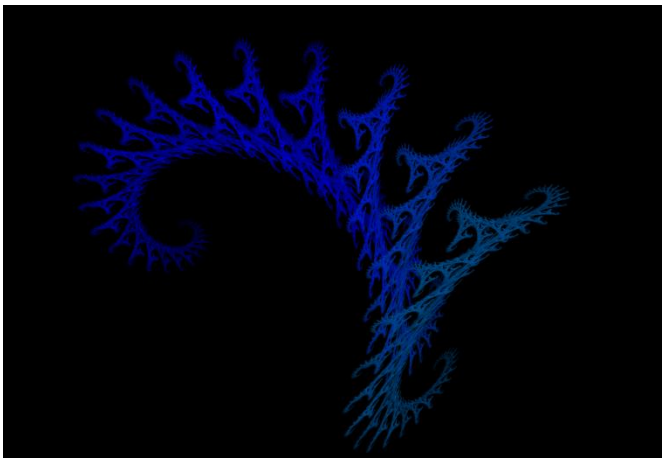
$$f_i : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$(x; y) \longmapsto (a_i x + b_i y + c_i;\ d_i x + e_i y + f_i) \qquad (a_i, b_i, c_i, d_i, e_i, f_i) \in \mathbb{R}^6$$

Toutes ces fonctions pour être contractantes doivent vérifier la propriété suivante :

$$(a - 1) \times (d - 1) - bc \neq 0$$

Chacune de ces fonctions admet un point fixe que la fonction ne modifie pas.



Flamme

Les fractales flammes sont issues des fractales IFS, en effet elles ne sont que des déformations de ces dernières. En plus d'une transformation par une des fonctions du système IFS, à chaque itération le point est en plus transformé par une fonction non-linéaire appelée variation flamme. Cette variation permet de déformer les motifs des fractales IFS selon des modèles bien précis, il existe déjà de nombreuses variation prédéfinies.

Exemple : Spherical, appliquée à l'IFS de la précédente illustration

$$f_i : \mathbb{R}_*^2 \rightarrow \mathbb{R}_*^2$$

$$(x; y) \longmapsto V_2(a_i x + b_i y + c_i;\ d_i x + e_i y + f_i) \qquad (a_i, b_i, c_i, d_i, e_i, f_i) \in \mathbb{R}^6$$

$$V_2(x; y) = \left(\frac{x}{\sqrt{x^2 + y^2}}; \frac{y}{\sqrt{x^2 + y^2}}\right)$$

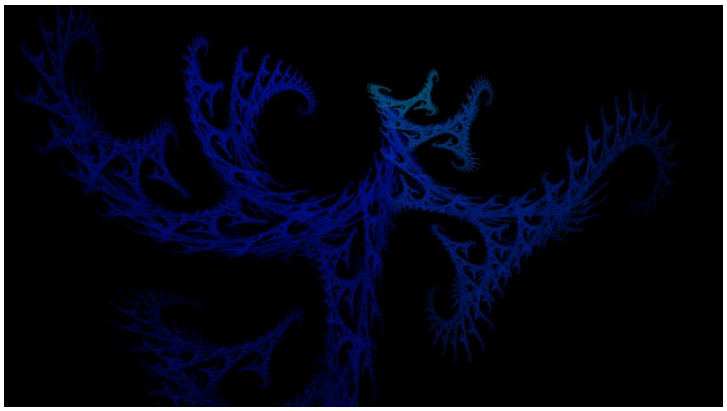


Image 5 : Transformation flamme "Spherical" appliquée à l'IFS de la précédente illustration

Description du programme

Fonctionnement général

Notre programme fonctionne sur le même principe pour le dessin et le calcul de toutes les fractales. Dans un premier, il analyse la ligne de commande (ou le menu), puis en fonction de la fractale choisie, lance une fonction initialisant les attributs de la structure correspondante. Ensuite, il lance un nombre de threads prédéfinis ayant pour objectif de calculer et de dessiner la fractale. Lorsque tous les threads sont terminés, le programme rafraichit l'écran, et lance une fonction d'attente d'événement déclenché par l'utilisateur. Lorsqu'un tel événement est déclenché, il est intercepté par la fonction de traitement des événements qui va amorcer une fonction appropriée en charge de répondre à l'action initiée par l'utilisateur. Cette action peut être de différents ordres, relancer un nouveau thread de dessin de la fractale avec des paramètres différents (zoom, déplacement, coloration, nombre d'itérations), augmenter le nombre de points sans rafraichir l'écran (pour les fractales de type Flamme et IFS), quitter le programme…

Ligne de commande et menu

L'interface utilisateur de notre programme est de type « ligne de commande » (CLI : command line interface). Sous le prompt d'une console Unix l'utilisateur entre la commande « ./Fractale.exe » qui accepte un ensemble d'options et de paramètres. Le detail en est donné en lançant la commande :

./Fractale.exe –h ou ./Fractale.exe --help

Une option « --full » permet d'obtenir une représentation graphique plein-écran.

L'ordre de passage des options et paramètres est sans importance. Certaines options réclament un paramètre, exemple « -w » doit être suivie d'une valeur indiquant la largeur de la fenêtre graphique en pixels. Toutefois l'omission de ce paramètre n'est pas fatale. Nous appliquons dans ce cas une valeur par défaut. C'est un choix d'implémentation. Nous aurions pu arrêter l'exécution et afficher l'aide pour que l'utilisateur corrige son oubli.

L'absence de certains paramètres obligatoires entraine, elle l'affichage d'un menu qui va permettre à l'utilisateur de compléter les instructions de lancement du programme ou d'interrompre l'exécution. Entrez sous le prompt Unix la commande « ./Fractale.exe » affiche ce menu.

Ce menu, permet entre autre de choisir parmi près de 30 transformations de fractales flamme, mais n'offre la totalité des options fournies en ligne de commande. Il faut le voir plus comme une aide pour un utilisateur « débutant ». Un utilisateur plus confirmé pourra spécifier directement à l'appel de la commande tout les paramètres ou options.

Mandelbrot et Julia

Pour tracer les fractales de Mandelbrot et de Julia, l'utilisateur peut passer par la ligne de commande (en utilisant l'option « -m » ou « -j ») ou par le menu. Lancer ces fractales par la ligne de commande lui permettra néanmoins de définir préalablement plus de paramètres tels que le zoom, la taille de l'écran…

Après avoir analysé cette ligne de commande, le programme lance 100 threads de dessins de la fractale Mandelbrot (50 pour l'ensemble de Julia). Chaque thread utilise une structure Mandelbrot ou Julia préalablement initialisée, contenant deux attributs consacrés à définir une abscisse de calcul de départ, et une de fin. Ainsi chaque thread parcourt, calcule et dessine les points d'un centième (deux centièmes pour Julia) de l'écran.

Une fois tous les threads terminés, le programme lance une fonction de gestion d'événement, qui attend et analyse les événements déclenchés par l'utilisateur (cf. Fonctionnement Général).

Ces différentes actions possibles pour Mandelbrot et Julia sont précisément indiquées dans le fichier « README ».

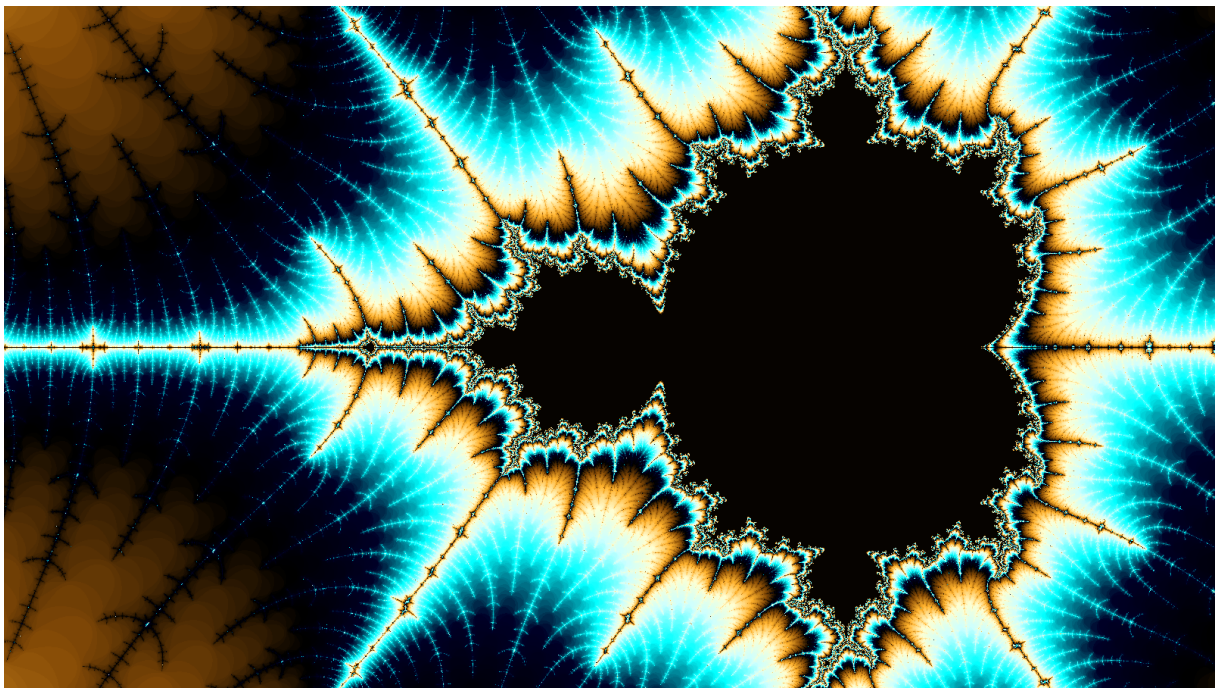


Image 6 : Fractale de Mandelbrot

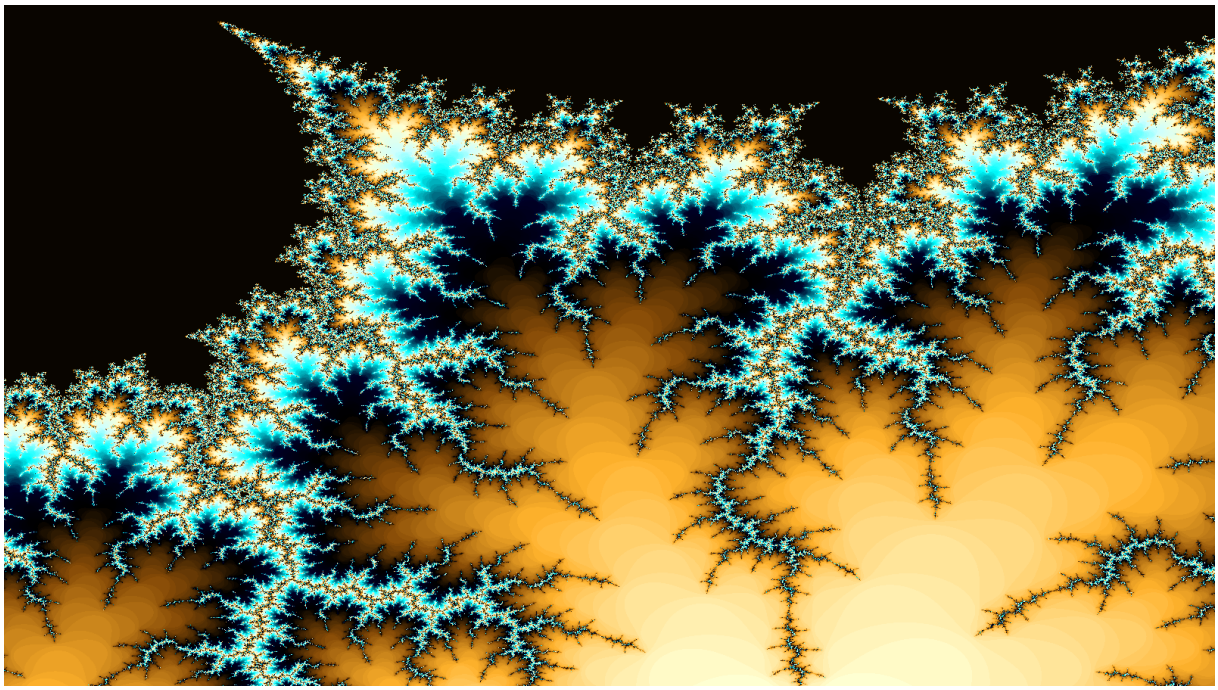


Image 7 : Fractale de Mandelbrot

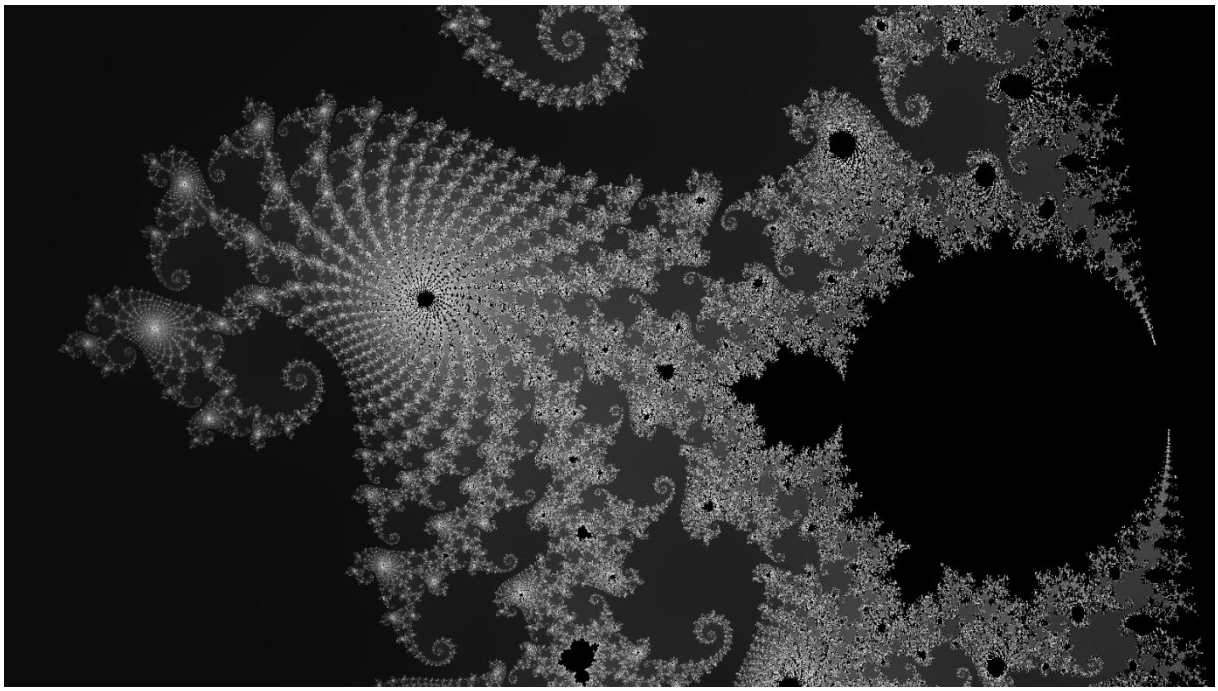


Image 8 : Fractale de Mandelbrot, noir et blanc

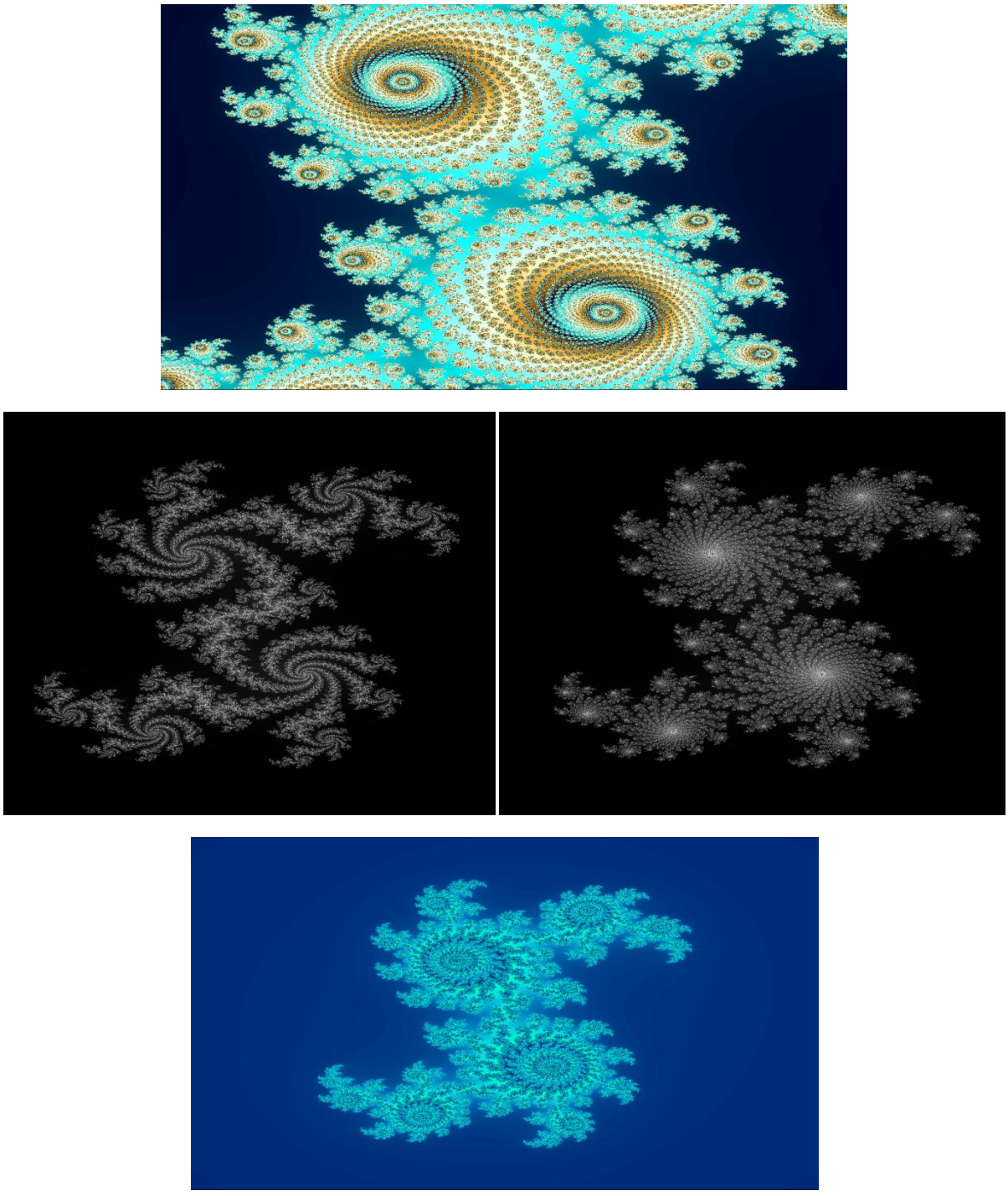


Image 1 : Différents ensembles de Julia, coloration multiples

L-System

Pour tracer les fractales de type L-System, l'utilisateur passera l'option « -l » à la commande et pourra optionnellement préciser un fichier de définition L-System à l'aide de l'option « -fl <LSystemfileName>» ou par le menu, auquel cas une fractale L-System par défaut sera dessinée.

Après avoir analysé cette ligne de commande, le programme initialise une structure correspondant au fractales de type L-System et lance 1 thread de calcul et de dessin de la fractale. Il analyse dans un premier temps le fichier, en extrait les règles, l'axiome, l'angle et la longueur, puis crée un tableau de caractères correspondant à la transformation à effectuer pour obtenir la fractale L-System à l'indice demandé. Dès que le tableau de caractères est crée, il est interprété par le programme pour dessiner la représentation graphique de la fractale. Là encore l'utilisateur peut interagir sur la représentation graphique.

Ces différentes actions possibles sur L-System sont précisément indiquées dans le fichier « README ». Leur gestion par le programme est quant à elle expliquée un peu plus loin dans le rapport.

Gestion des fichiers de type L-System

Les fichiers L-System sont contenus dans le dossier « ./files_config/lsystem/ », chacun de ces fichiers définit toute les propriétés caractérisant une fractale. En respectant ce format l'utilisateur peut se créer ses propres fichiers L-System. Chaque fichier décrit en fait une grammaire dont nous allons donner le détail.

Exemple de fichier L-System :

```
F~;
60
5
F
F+F--F+F
```

La première ligne du fichier décrit l'alphabet utilisé (F dans l'exemple ci-dessus). Le séparateur de motif de l'alphabet est le caractère « ; ».

Le caractère « ~ » suivant un motif de l'alphabet signifie que ce motif devra être tracé. Tout autre caractère sera interprété comme « ne pas tracer ».

La seconde ligne correspond à l'angle de rotation, cette angle peut être décimal et a pour unité le degré.

La troisième ligne correspond à la longueur du trait, cette longueur a pour unité le pixel.

La quatrième ligne correspond à l'axiome, c'est-à-dire à la transformation de départ. Notre phrase dans le paragraphe L-System de l'introduction.

Les lignes suivantes correspondent aux règles de transformations associées aux motifs de l'alphabet définis sur la première ligne (dans l'exemple : `F+F--F+F`). Chaque motif F sera transformé en `F+F--F+F`. Il doit donc y avoir autant de règles que de motifs dans l'alphabet. Les règles de transformations peuvent utiliser les caractères suivant :

- \+ : rotation horaire de l'angle défini à la seconde ligne du fichier
- \- : rotation horaire de l'angle défini à la seconde ligne du fichier
- * : rotation de 180°
- [: sauvegarde la position en cours
-] : restaure une position

Le fichier L-System est analysé par le programme qui renvoie une erreur en cas d'inconsistance dans la description de la grammaire et/ou de non respect du dictionnaire utilisé pour décrire les règles de transformations.

Si l'analyse du fichier L-System est un succès, le programme stocke les informations contenues dans les lignes dans deux tableaux de chaines de caractères. Le premier contenant les informations telles que l'angle et la longueur, le second contenant les noms des variables et les règles associées.

Sur la page suivante quelques résultats graphiques obtenus par analyse de fichiers du répertoire files_config/lsystem/

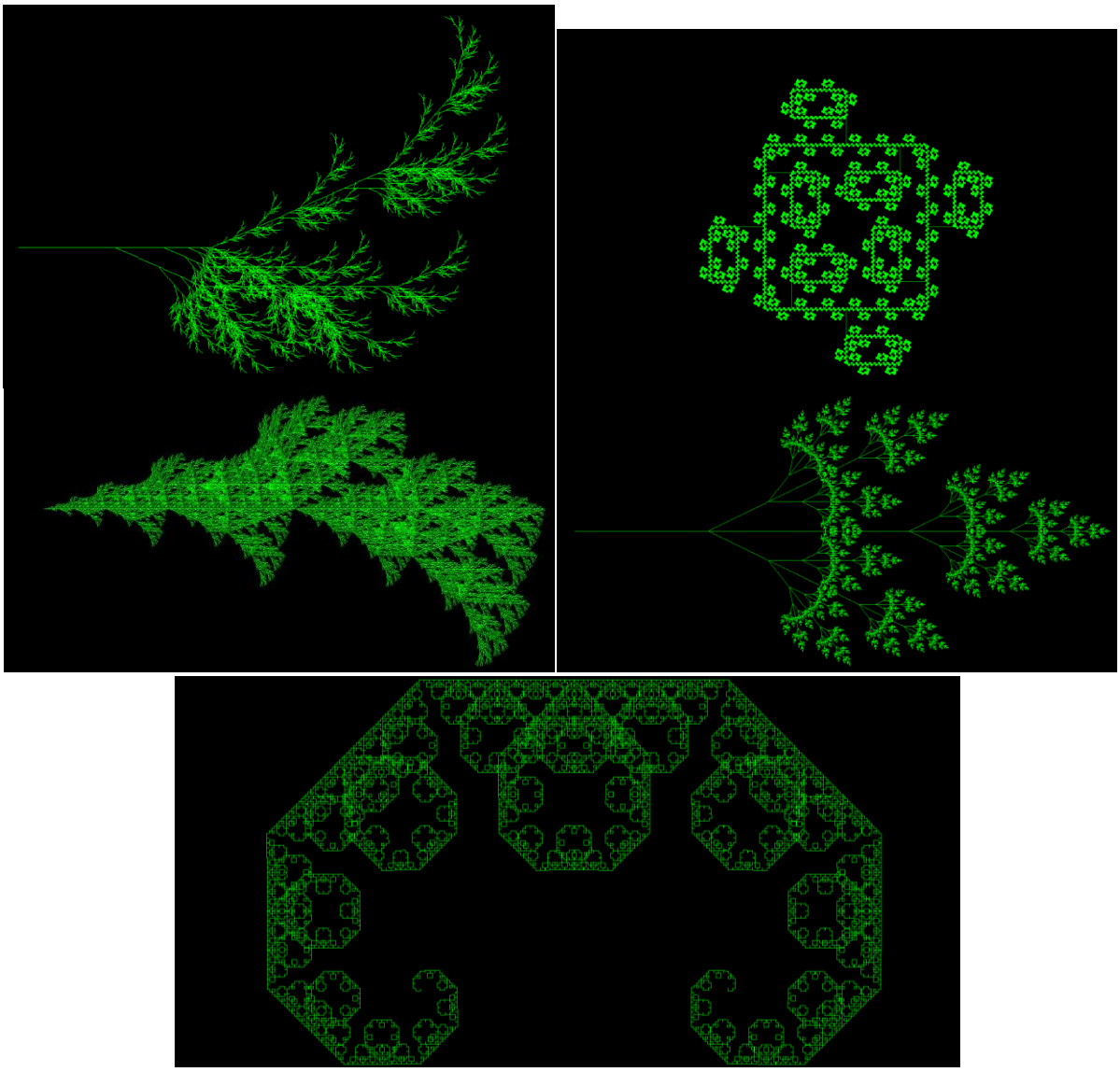


Image 9: Différentes fractales de type L-System

IFS

Pour tracer les fractales de type IFS, l'utilisateur passera l'option « -i » à la commande et pourra optionnellement préciser un fichier de définition IFS à l'aide de l'option « -fi <IFSfileName>» ou par le menu auquel cas une fractale IFS par défaut sera dessinée.

Après avoir analysé cette ligne de commande, le programme initialise une structure correspondant aux fractales de type IFS et lance 1 thread de calcul et de dessin de la fractale. Il analyse dans un premier temps le fichier associé, en extrait les 6 coefficients caractérisant une fonction IFS ainsi qu'un septième attribut correspondant à la probabilité d'appel de la fonction.

Une fois que la fractale est dessinée le programme lance une fonction de gestion d'événements, qui attends et analyse les événements déclenchés par l'utilisateur

Ces différentes actions possibles pour les fractales de type IFS sont précisément indiquées dans le fichier « README.txt ». Leur gestion par le programme est quand à elle indiquée un peu plus loin dans le rapport.

Gestion des fichiers de type IFS

Différents exemples de fichiers flamme sont disponibles dans le dossier « /files_config/ifs/ ». Il est possible d'en créer soi-même.

Exemple de fichier IFS : Floor.ifs

```
0.0 -0.5 0.5 0.0 -1.732366 3.366182 0.333333;
0.5 0.0 0.0 0.5 -0.027891 5.014877 0.333333;
0.0 0.5 -0.5 0.0 1.620804 3.310401 0.333334
```

Un fichier peut être composé de n lignes, n étant le nombre de fonctions IFS. Chaque ligne doit se présenté sous la forme suivante :

$$a\ b\ c\ d\ e\ f\ p\ ;$$

Où a, b, c, d, e et f sont les coefficients de la fonction IFS tels que :

$$f(x;y) = (ax + by + e;\ cx + dy + f)$$

p représente la probabilité d'appel de cette fonction, la somme des probabilités de toutes les lignes doit être égale à 1.

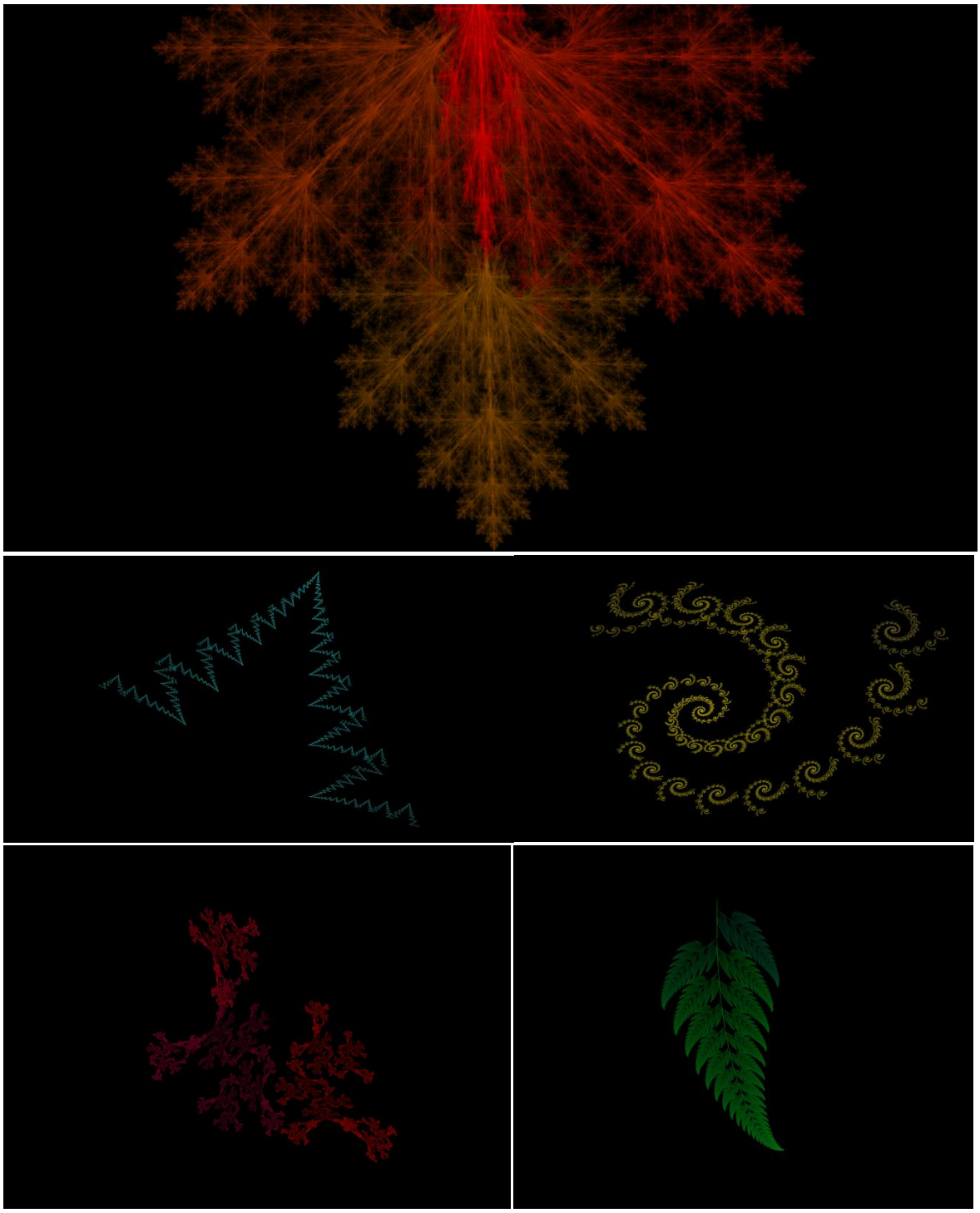


Image 10 : Fractales de type IFS

Flamme

Pour tracer les fractales de type Flamme, l'utilisateur peut passer par la ligne de commande ou par le menu. Le menu lui proposera de choisir parmi près d'une trentaine de variations flammes prédéfinies associées à une dizaine d'IFS, soit environ 300 fractales flamme prédéfinies. Néanmoins, le programme contenant un analyseur syntaxique d'expressions mathématiques, il est possible de créer ou de tester soit même des variations flammes en les écrivant soi-même dans un fichier du répertoire « /files_config/flame/ ». Les expressions n'étant dans ce cas pas prédéfinies, le programme doit évaluer lui-même à l'aide de l'analyseur syntaxique d'expressions mathématiques les transformations indiquées dans le fichier, le dessin des fractales depuis cette méthode est donc près de 5 fois plus long que depuis le choix parmi les transformations prédéfinies dans le menu.

Après avoir analysé la ligne de commande, ou encore le choix effectué dans le menu, le programme initialise une structure correspondant aux fractales de type flamme et lance 1 thread de calcul et de dessin de la fractale. Il analyse ensuite le fichier associé et réalise les calculs de transformation Flamme sur chaque point Ifs calculé. Dans le cas de la méthode de choix par menu, le programme associe le choix à une fractale correspondant à une fonction prédéfinie dans celui-ci, et utilise donc cette fonction, ce qui réduit nécessairement le temps de calcul.

Une fois que la fractale est dessinée et que le thread est donc terminé, le programme lance une fonction de gestion d'événements, qui attend et analyse les événements de l'utilisateur. En fonction de ces événements, le programme lancera à nouveau un dessin avec des paramètres différents ou quittera le programme, suivant l'action de l'utilisateur.

Ces différentes actions possibles pour les fractales de type Flamme sont précisément indiquées dans le fichier « README ». Leur gestion par le programme est quant à elle indiquée un peu plus loin dans le rapport.

Gestion des fichiers de type Flamme

Différents exemples de fichiers flamme sont disponibles dans le dossier « /files_config/flame/ ». Il est possible d'en créer soi-même une infinité selon les règles suivantes ;

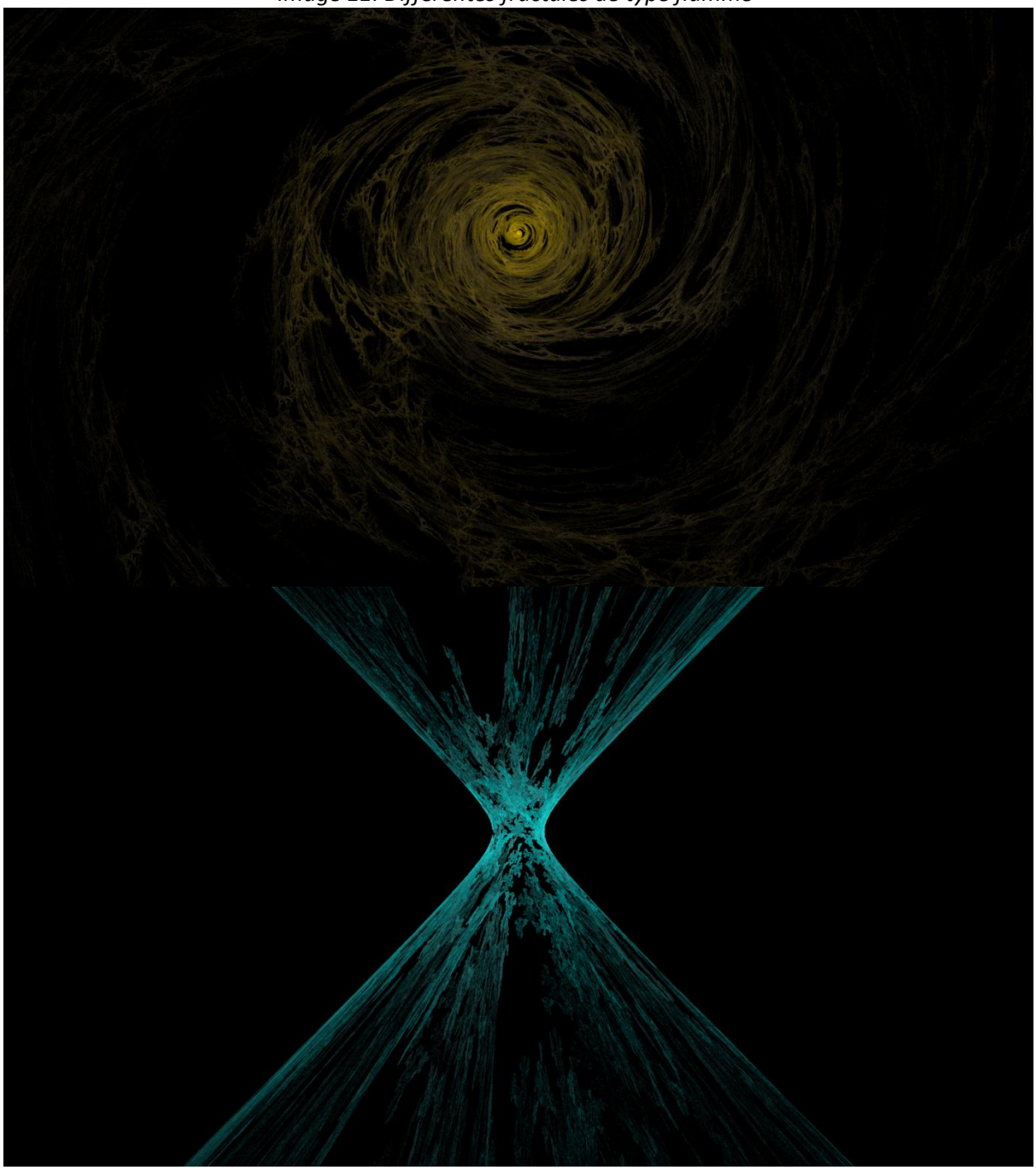
La première ligne du fichier correspond au nom du fichier Ifs associé à la transformation flamme (avec le chemin relatif ou absolu).

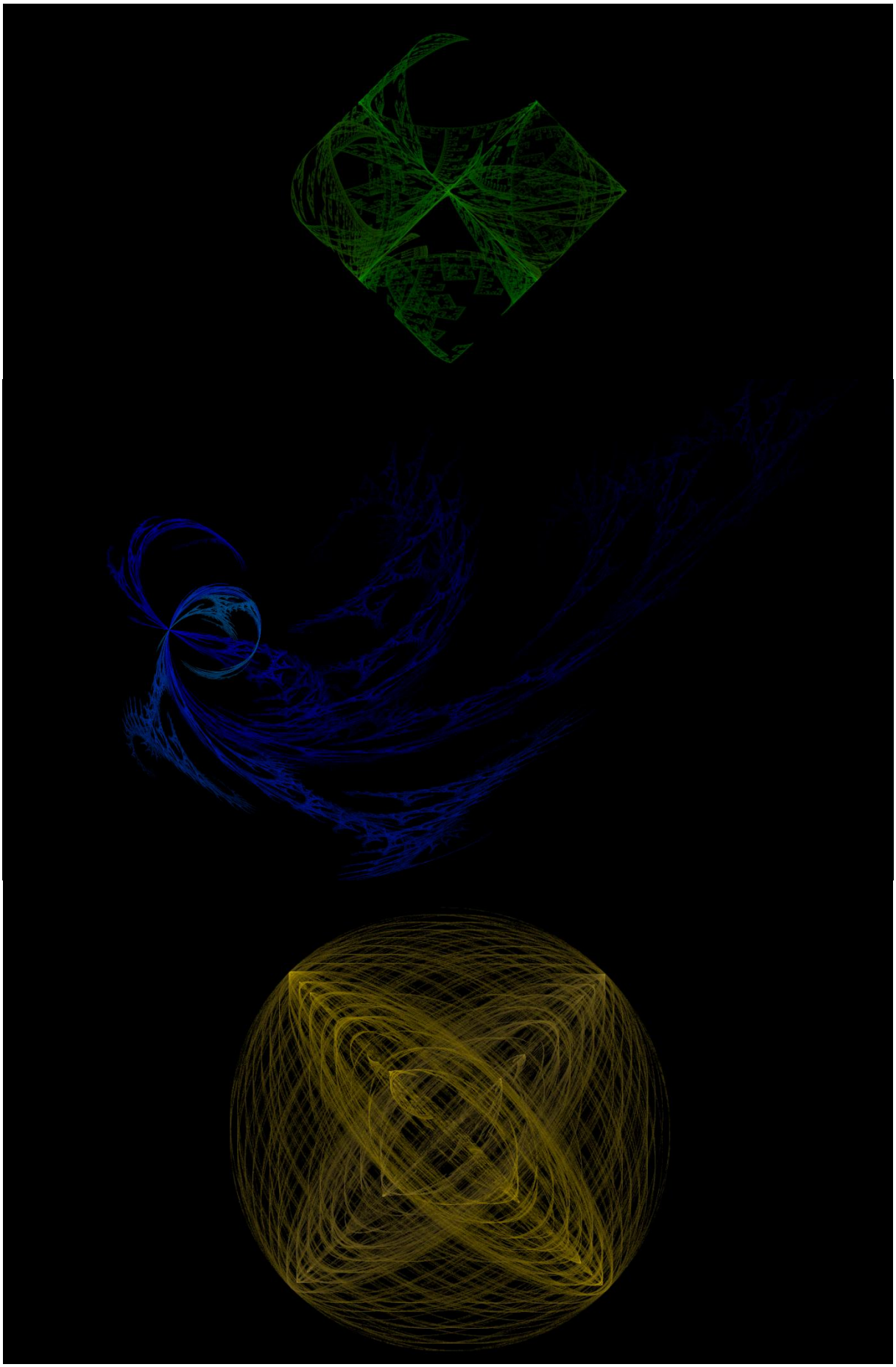
La seconde ligne du fichier correspond à l'expression de la transformation à associer à l'abscisse du point.

La troisième ligne correspond à l'expression de la transformation à associer à l'ordonnée du point.

Ces deux dernières lignes doivent respecter la syntaxe de l'analyseur et calculateur d'expression mathématique définie dans le paragraphe suivant ainsi que dans le fichier « README.txt ».

Image 11: Différentes fractales de type flamme





Analyseur syntaxique et calculateur d'expressions mathématiques

Afin d'analyser et de calculer les expressions mathématiques flammes rentrées dans les fichiers, nous avons implémentées un analyseur de syntaxe mathématique.

Le programme analyse tout d'abord si les expressions rentrées dans les lignes du fichier sont correctes, c'est-à-dire si tous les caractères présents sont connues, par exemple des chiffres, Pi, des opérateurs (tels que +, -, *, /, ^,%), des fonctions (cosinus, exponentielle…). En cas de caractère incorrect, il précise la ligne du fichier posant problème, ainsi que la position de l'erreur dans la ligne. De même, s'il y a une erreur liée aux parenthèses, le programme la détecte. Enfin, les erreurs d'écriture telles que des opérateurs se suivant comme « 4**2 » sont elles aussi détectées.

Le programme est aussi capable de gérer les nombres décimaux et les nombres négatifs, et donc de différencier l'opérateur « -» du signe « - », ainsi que d'analyser le symbole « . » comme la virgule. Enfin les espaces ne posent pas de problème, et la totalité des cas d'erreurs sont traités.

Après avoir analysé tous les caractères et la cohérence de l'expression mathématique, et donc après avoir vérifié que celle-ci était correcte, le programme convertit l'expression infixe écrite par l'utilisateur en une expression postfixe afin de se débarrasser des parenthèses, et de rendre la complexité du calcul plus faible lors de l'utilisation de l'expression pour calculer les coordonnées des points lors de l'application de la transformation flamme. En effet, l'algorithme de calcul d'une expression postfixée est plus rapide que celui d'une expression infixe.

Lors du dessin d'une fractale Flamme, cette partie d'analyse et de transformation de l'expression ne s'effectue qu'une fois. Le programme travaille ensuite avec deux expressions postfixes correspondant à la transformation flamme associée à l'abscisse et à l'ordonnée du point. Durant le calcul des points Flamme, le programme effectue un algorithme de calcul d'expressions postfixes utilisant une pile. Toutefois, les expressions infixes restent tout de même plus longues à analyser qu'en mode menu, ou le programme n'a qu'a rentré dans un « switch » lançant une fonction contenant le calcul associé.

Les différentes fonctions et constantes reconnues par l'analyseur syntaxique sont indiquées dans le fichier «README.txt »

Le lissage de la coloration

Mandelbrot et Julia

Pour effectuer un dégradé de couleur relativement lisse à l'extérieur de l'ensemble de Julia et de Mandelbrot, nous avons défini un taux de divergence compris entre 0 et 1 qui est le rapport entre le nombre d'itérations effectuées avant que le module ne dépasse 2 avec le nombre maximum d'itérations.

Ensuite nous avons créé un tableau contenant plusieurs couleurs, la première correspondant à la couleur des points dont le module dépasse 2 dès la première itération alors que la dernière représente la couleur des points dont le module dépasse 2 à partir d'un nombre d'itérations proche de celui maximum. En effet les points dont le module ne dépasse pas 2 avant d'atteindre le nombre maximum d'itérations sont noir.

Par exemple pour un point dont le module dépasse 2 après 75 itérations sur 300 au maximum, sa couleur est composée à 50% de bleu et à 50% de blanc.

Nous avons par la suite agrandi ce tableau en répétant plusieurs fois le même dégradé, ce qui permet également de retrouvé plusieurs fois les couleurs dans les ensembles de Julia et de Mandelbrot.

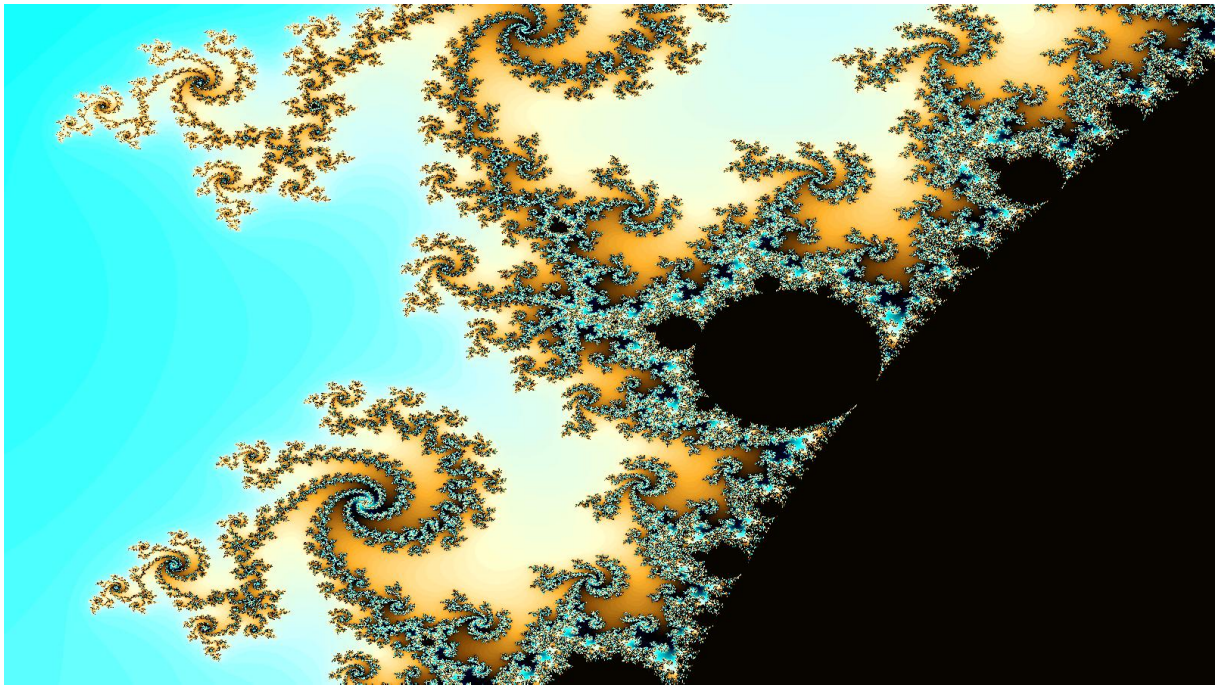


Image 12 : Lissage sur Mandelbrot

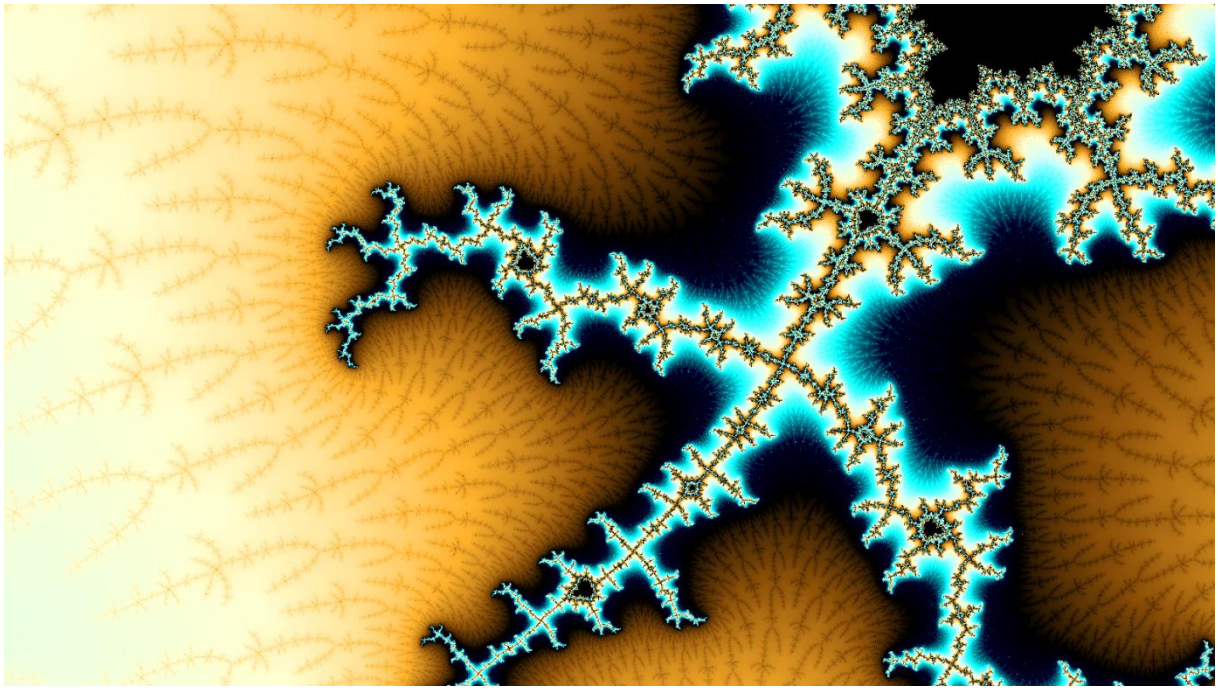


Image 13 : Lissage sur Mandelbrot

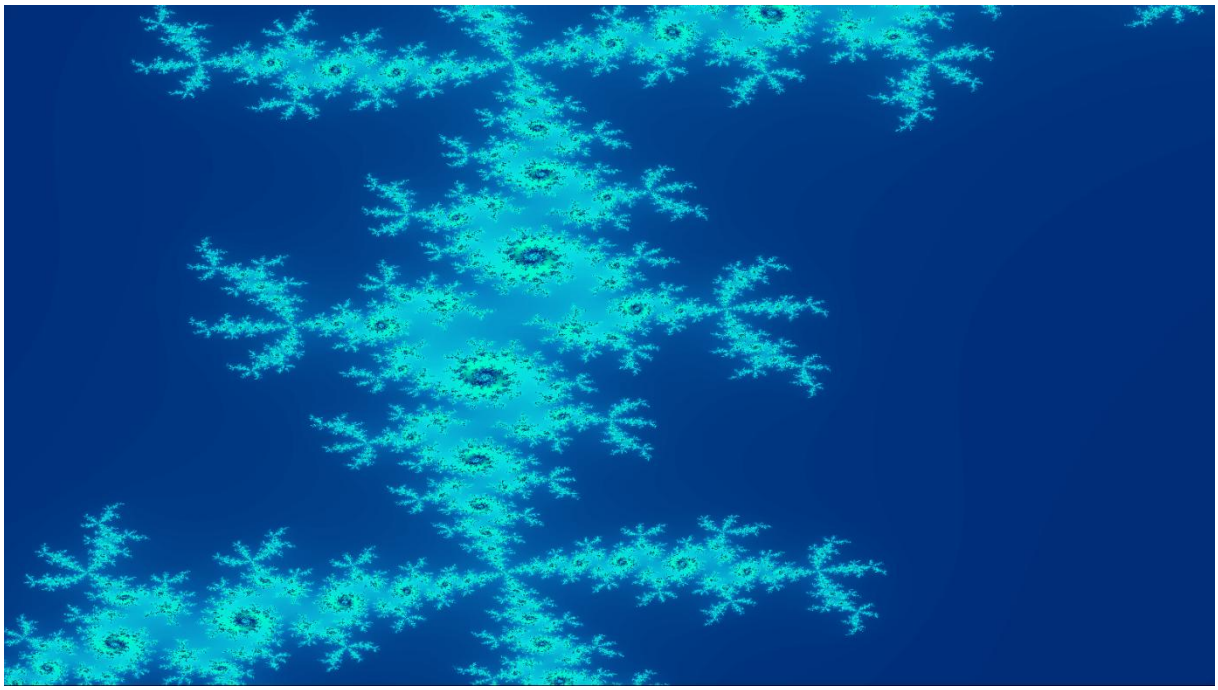


Image 14 : Lissage de la coloration sur un ensemble de Julia

Ifs et Flamme

Afin d'effectuer au mieux un lissage de la coloration sur les fractales IFS et flamme, nous avons utilisé un algorithme qui dans un premier temps lors du calcul des points IFS ou flamme, incrémente la valeur d'un tableau deux dimensions représentant l'écran lorsqu'un point est calculé. Par exemple, si le point calculé a pour coordonnées (x,y), alors on incrémente la valeur du tableau dont la première dimension correspond à l'abscisse x, et la seconde à l'ordonnée y. Chaque case du tableau contient le nombre de fois où le point correspondant à été retourné par une des fonctions IFS.

Après avoir calculé les points et donc rempli le tableau représentant l'écran, on va dessiner la fractale sur l'écran. Pour ce faire, on parcourt le tableau deux dimensions, et on lui applique la couleur correspondant au logarithme de la valeur du tableau divisé par le logarithme de la valeur maximale du tableau, multiplié par 255.

$$\textbf{couleur} = \frac{\log(\text{tab}[x][y] + 1)}{\log(\text{maxTab} + 1)} \times 255$$

La coloration est établie à partir du cardinal lu dans chaque case et de la formule ci-dessus. On assiste ainsi à un lissage de la coloration en fonction du nombre de fois ou un point à été calculé, le point ayant été calculé le plus étant le plus vif.

A ce mode de coloration, nous avons ajouté la possibilité de donner une couleur à chaque fonction IFS.

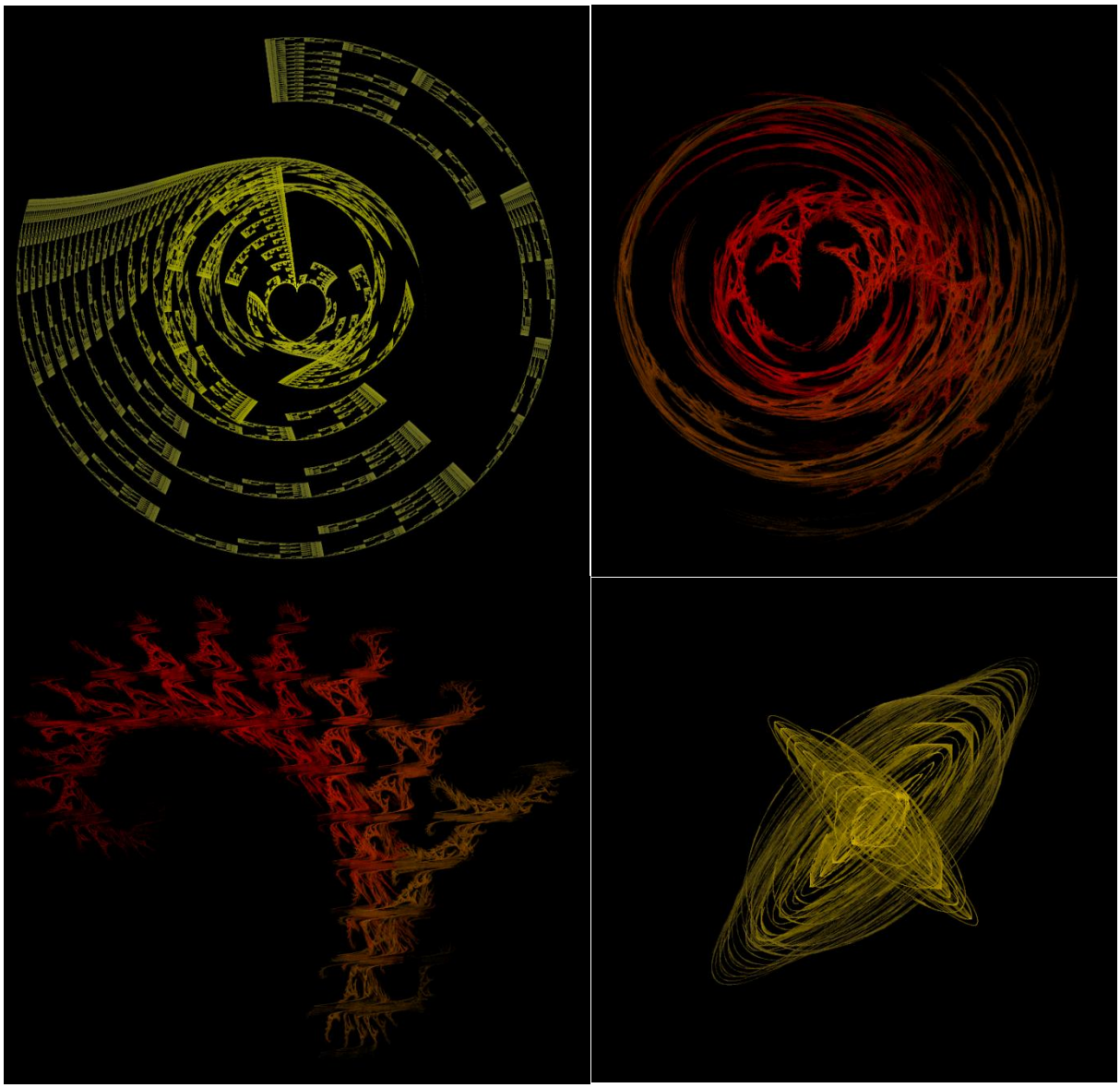


Image 15 : Lissage sur des fractales de type flamme

Mise à jour en temps réel des ensembles de Julia

La mise à jour en temps réel des ensembles de Julia s'effectue suivant les coordonnées du pointeur de la souris. Les coordonnées de la souris sont transformées en deux réels compris entre 0 et 1 correspondant au complexe C de la fractale de Julia (la coordonnée x correspond à la partie réel et y à la partie imaginaire).

Afin d'accentuer la fluidité, et outre l'utilisation des threads, nous avons utilisé la fonction SDL_PoolEvent permettant de retirer tous les évènements de la file d'événements tant que l'événement en cours n'est pas totalement traité.

Calcul de l'ensemble de Julia via itération inverse

La méthode de calcul de Julia via itérations inverses, est un calcul de racine carré complexe. Comme pour la méthode de Julia une constante complexe C est fixée et pour chaque pixel de l'image un Z_0 est attribué. Il faut ensuite itérer en choisissant aléatoirement à chaque itérations une des deux fonctions de récurrence suivante:

$$Z_{n+1} = \sqrt{Z_n - C}$$

$$Z_{n+1} = -\sqrt{Z_n - C}$$

A chaque itération il faut dessiner (sauf pendant les 20 premières, temps de stabilisation) le point Z_{n+1} calculé. Pour la version du calcul par itérations inverses en temps réel la constante C correspond aux coordonnées de la souris transformées en complexes.

La touche « c » du clavier permet de passer en mode itération inverse sur une représentation graphique de Julia.

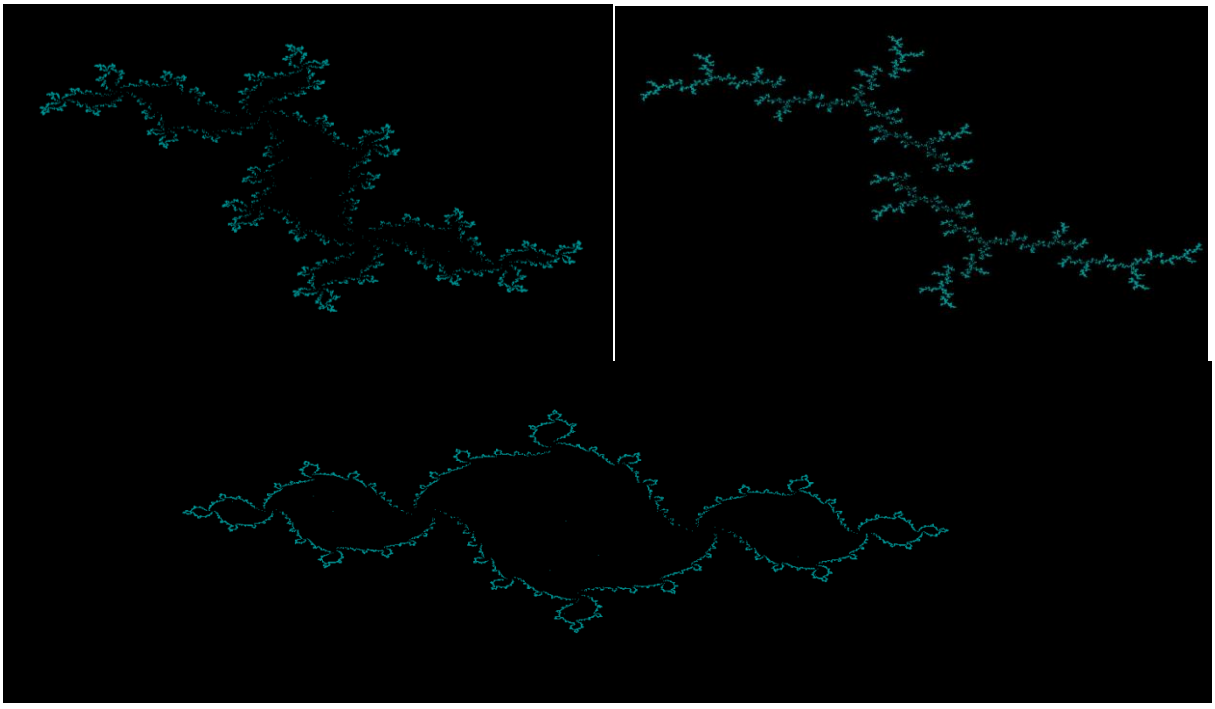


Image 16: Ensemble de Julia par itérations inverse

Actions et événements

Pour chaque fractale, l'utilisateur a le choix d'effectuer différentes actions, sur la représentation graphique à l'aide du clavier (et/ou pour les calculs en temps réel à l'aide de la souris) interprétés et gérés par le programme

Déplacement (flèches du clavier)

Disponible sur toutes les fractales, permet de changer la position de la fractale. Chaque structure possède un pointeur vers un point initial et ou déplacement, le programme modifie donc les coordonnées du point de déplacement ou initial, afin de modifier la position de la fractale.

Zoom et dézoom (page Up/Down)

Disponible sur toutes les fractales, permet de se rapprocher ou de s'éloigner de la fractale. Chaque structure représentant un type de fractale possède un attribut correspondant à un pointeur vers le zoom, la valeur pointée est donc modifié pour zoomer ou dézoomer.

Capture d'écran (F5)

Disponible sur toute les fractales, le programme enregistre une image représentant l'écran (sous format Bitmap) dans le dossier « screenshots » prévu à cet effet. C'est depuis cette fonction qu'ont été réalisées la totalité des images de ce rapport. La résolution de l'image correspond à la taille de l'écran lors de la capture d'écran.

Augmenter/réduire le nombre d'itérations ou de points (+,-)

Disponible sur toutes les fractales au travers du nombre d'itérations pour les fractales de Mandelbrot, de Julia et L-System, et du nombre de points pour les fractales Ifs et Flamme. Pour augmenter ou réduire le nombre d'itérations, on redessine la fractale en modifiant le nombre d'itérations, attribut de la structure représentant la fractale. Il en est de même pour la diminution du nombre de points, néanmoins, grâce à la gestion des threads, augmenter le nombre de point sur Ifs ou Flamme ne nécessite pas de redessiner la totalité de la fractale avec plus de point, mais juste de relancer un thread de dessin avec autant de point sans effacer l'écran.

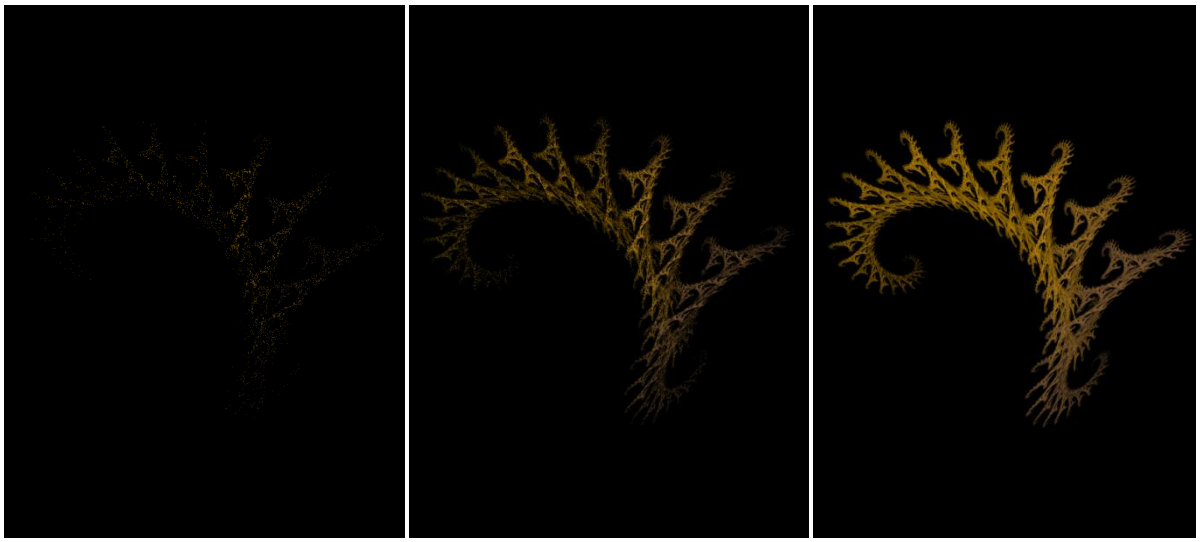


Image 17 : Augmentation du nombre de points sur une fractale de type IFS

Modification de la coloration (espace)

Disponible sur toutes les fractales sauf L-System. Différents tableaux de dégradés ou de couleurs sont disponibles pour ces fractales, ainsi les différentes structures associées aux fractales possèdent un attribut responsable de la coloration.

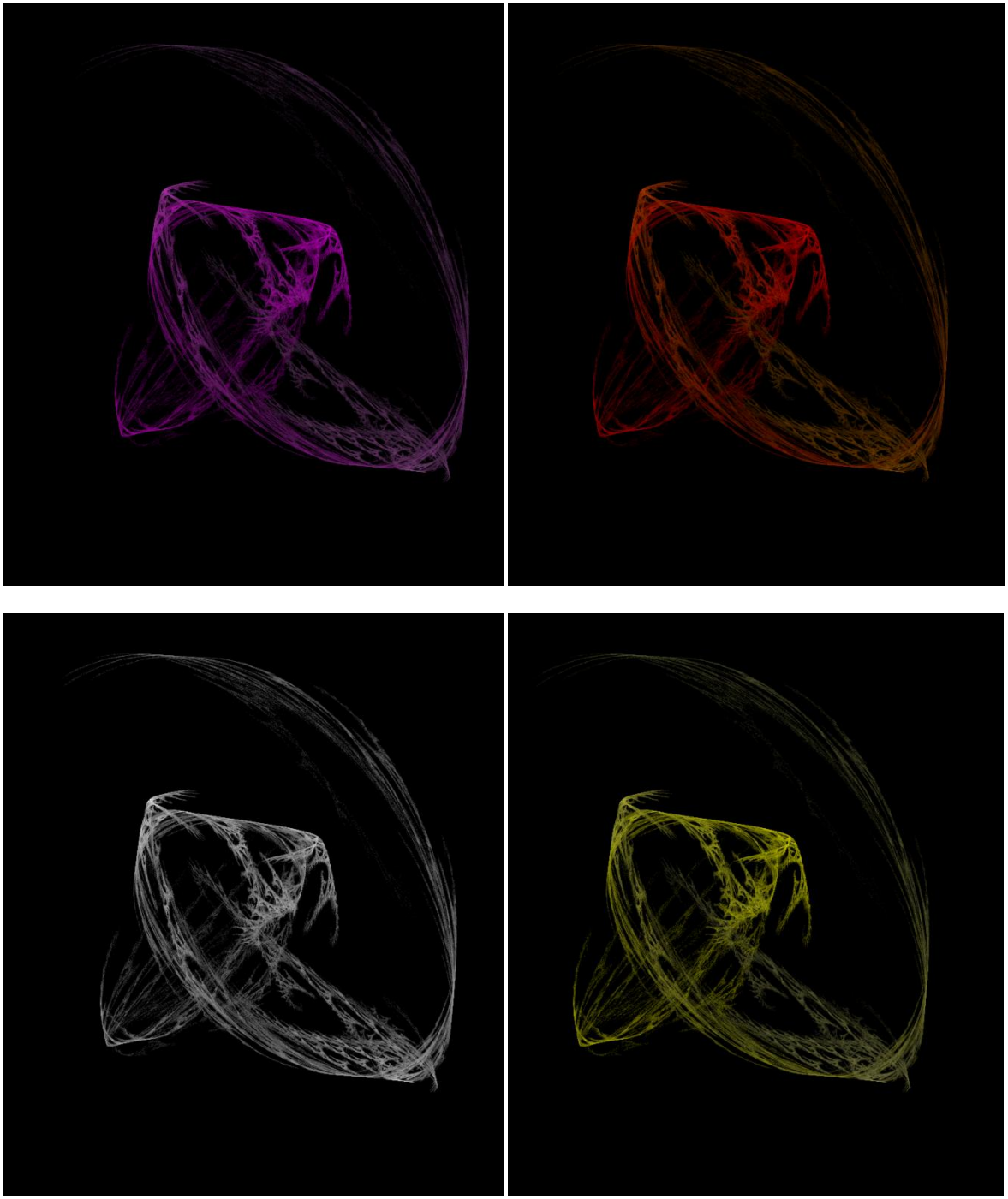


Image 18 : Différentes colorations d'une fractale de type flamme

Changement de mode de coloration (touche c)

Disponible uniquement sur les fractales Ifs et flamme. Permet de passer d'un mode monochromatique à un mode de coloration ou chaque fonction Ifs est associée à une couleur.

Contrôle de l'intensité des couleurs (crtl +, ctrl -)

Disponible sur les fractales de type Ifs et flamme, permet d'augmenter ou de réduire l'intensité des couleurs, en les associant avec plus ou moins de noir. Ici alpha représente l'intensité de la couleur initiale (comprise entre 0 et 1)

$$\text{couleurFinale} = \text{noir} \times (1 - \boldsymbol{\alpha}) + \boldsymbol{\alpha} \times \text{couleurInitiale}$$

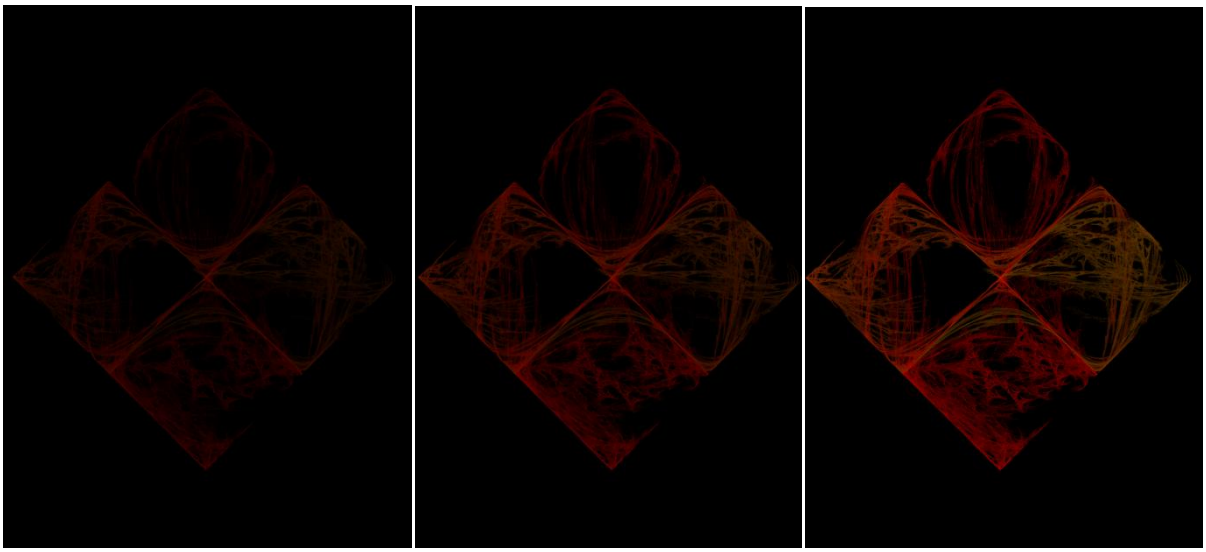


Image 19 : Contrôle de l'intensité des couleurs sur une fractale de type flamme

Multiprocesseur

L'un des facteurs auquel sur lequel nous avons été très attentif est la vitesse de dessin des fractales, voilà pourquoi nous nous sommes intéressé aux threads et à l'intérêt que pouvait avoir ceux-ci, notamment lors de l'utilisation de processeurs capables de gérer simultanément plusieurs processus. Ainsi, pour dessiner Mandelbrot et Julia, le programme lance respectivement 100 et 50 threads de calcul, chacun traitant une partie de l'affichage.

Problèmes rencontrés

Mandelbrot et de Julia

Sur Mandelbrot et Julia, les principaux problèmes rencontrés étaient relatifs aux performances graphiques. Comment faire tourner un maximum d'itération en un temps minimum? Ce fût notre principale problématique. Pour y répondre, nous avons utilisé deux moyens qui ont réellement fait leurs preuves.

- le multithreading (70% de gain en vitesse d'affichage).
- Un algorithme se basant sur le fait qu'en général, deux pixels côte à côte possèdent la même couleur. Cet algorithme est expliqué dans la partie « Avantage du programme ».

L-System

Pour les fractales de type L-System, nous avons du commencer à effectuer des allocations mémoires dynamiques, créer une structure pile… Des notions vu en CPI1 mais à propos desquelles il ne nous restait que très peu de souvenirs. Nous avons donc eu au départ quelques difficultés d'allocations mémoire, problèmes qui sont peu à peu devenus évidents à résoudre au fur et à mesure que nous progressions et que nous comprenions le fonctionnement du langage C et des pointeurs.

C'est aussi pour L-System que nous avons du utiliser pour la première fois les différentes méthodes de lecture fichier. Il fallait pouvoir lire différentes informations dans un même fichier, sans en connaitre le nombre (le nombre de règles pouvant être illimité).

IFS

La difficulté majeure de la partie IFS, a été en tout premier lieu la compréhension de la méthode mathématique et algorithmique du problème. La compréhension du jeu du chaos, et surtout le fait qu'un appel de fonctions aléatoire et désordonné pouvait donner une forme aussi précise. Une fois la méthode assimilée l'implémentation n'a pas révélé de problèmes particuliers.

Flamme

L'unique difficulté pour le dessin des fractales de type flamme résidait dans le fait de devoir lire dans un fichier, des expressions mathématiques dont on ne connaissait ni la longueur, et ni les fonctions utilisées. Il fallait donc trouver le moyen de identifier les fonctions mathématiques, de différencier les opérateurs des opérandes, d'analyser les priorités, les parenthèses… Après plusieurs essais de réalisations d'algorithmes permettant d'analyser une expression, nous nous sommes rendu compte que ce qui posait le plus de problèmes était la gestion des parenthèses. C'est pour cette raison que nous avons décidé d'appliquer à l'expression mathématique une transformation en notation polonaise (notation infixe). Nous appliquons ensuite l'algorithme de calcul d'une expression infixe en utilisant les piles.

De nombreuses autres difficultés ont aussi été rencontrées lors de l'analyse de l'expression à calculer. Tout d'abord, il était nécessaire de mettre à disposition à l'utilisateur du programme des fonctions indispensables et très utilisées pour le dessin des fractales de type flamme, telles que cosinus, sinus, tangente, arc cosinus, arc sinus, arc tangente, exponentielle ou encore le calcul direct du module. De même, il était intéressant de mettre à disposition la constante Pi. Ensuite, il a fallut gérer les nombres décimaux dans l'analyse de l'expression, la difficulté est en effet de faire le lien entre des caractères se suivant, et de calculer ce qu'ils représentent. De même, la gestion des nombres négatifs à été complexe dans le sens où il ne fallait pas que l'analyseur d'expression confonde le signe et l'opérateur «-».

Les fuites mémoire

Lorsque nous avons programmé, nous ne nous sommes quasiment pas penchés sur les fuites mémoire, et cela nous a forcément posé un problème lorsque, quelques semaines avant la date de rendu du projet, nous avons décidé d'évincer toute fuite mémoire de notre programme.

Nous avons donc utilisé le logiciel Valgrind, détectant les erreurs mémoires, qui nous a été très utile même si la lecture et la compréhension des erreurs que celui-ci indiquait peut s'avérer assez difficile, notamment lors des premières utilisations, où la notion de « jump » par exemple, nous était inconnue.

L'élimination des fuites mémoires nous a obligé à modifier quelque peu notre code, et il est évident que nous aurions du les gérer parallèlement à l'avancement du code.

Avantages du programme

Vitesse de dessin

Grâce à l'implémentation de différentes techniques, le dessin des fractales de Mandelbrot et Julia est près de 4 fois plus rapide.

Threads

La vitesse de calcul et de dessin des fractales de Mandelbrot et de Julia, de part la répartition de la surface à traiter entre de nombreux threads, est un avantage certain quand à la vitesse du programme. Le gain de temps s'élève aux alentours de 70% en moyenne avec un micro-processeur 4 cœurs. Ceci dit, seuls les micro-processeurs récents (2 cœurs et plus) trouveront réellement un avantage dans l'implémentation des threads.

Algorithme de calcul des fractales de Mandelbrot et Julia

Afin de gagner du temps sur les dessins des fractales de Julia et Mandelbrot, nous avons implémenté un algorithme se basant sur le fait qu'assez souvent, deux points côte à côte possèdent la même couleur. Ainsi, on si un pixel de coordonnée (x,y) possède la même couleur qu'un pixel de coordonnée (x,y+2), alors la couleur du pixel de coordonnée (x,y+1) n'est pas calculée mais supposée identique à celle des points l'encadrant. Cette algorithme nous as encore permit de gagner 47% de temps en moyenne.

Analyseur syntaxique

L'analyseur syntaxique d'expressions mathématiques est un avantage important dans le sens ou il permet de créer une infinité de transformations flamme, combinant des dizaines de fonctions mathématiques disponibles.

Fuites mémoires et erreurs de segmentation

Durant les dernières semaines, nous avons supprimé toutes les fuites et les erreurs mémoires du programme grâce au logiciel Valgrind, ce qui ne peut être que bénéfique pour un programme qui réalise tout de même de nombreuses allocations mémoires, et qui peut être amené à exploiter énormément de ressources. De même, nous avons fait en sorte qu'il ne puisse exister d'erreur de segmentation, notamment lors des lectures de fichier qui quelque soit leur syntaxe ne renverront pas d'erreur de segmentation.

L-System

Pour les fractales de type L-System, l'utilisateur peut créer des fractales avec une infinité de règles, et décidé pour chacune s'il souhaite qu'elle trace ou non, contrairement à une implémentation basique où les règles concernant les variables de tracé sont prédéfinis dans le programme.

Plein écran et capture d'écran

Ces deux options que nous avons mises à disposition de l'utilisateur permettent d'obtenir des images haute résolution des fractales, et ainsi les utiliser à des fins diverses (fond d'écran…).

Lancement de la commande

Nous avons pris le parti de ne pas interrompre l'exécution du programme en cas d'omission de certains paramètres. Ceci pour ne pas pénaliser l'utilisateur. Des valeurs par défaut sont utilisées pour palier à l'absence d'un paramètre (-w sans spécification de pixel). Un menu s'affiche en cas d'omission du type de fractales à dessiner. Des fichiers de définition par défaut sont utilisées si l'option –f est utilisée sans spécification par l'option –ff d'un fichier Flamme etc. L'autre solution aurait été de présenter l'aide sur toute détection d'erreur dans le passage des options/paramètres et de quitter le programme.

Ce qui aurait pu être amélioré

Gain de temps

Même si la vitesse de calcul et de dessin du programme fait parti des avantages de celui-ci, nous aurions bien aimé passer un peu plus de temps sur certaines fractales afin de mettre en place des algorithmes permettant de gagner encore plus de temps. Ainsi pour Mandelbrot et Julia par exemple, nous pensons qu'il aurait été possible de gagner énormément de temps lors du zoom, ou du déplacement, en se servant des pixels calculés lors du précédent affichage. En effet, sur un zoom de coefficient 2, la moitié des points de la fractale a déjà été calculée précédemment. En stockant le nombre d'itérations lié à tous ces points déjà calculés dans un tableau deux dimensions représentant l'écran (tout comme on le fait sur les fractales flammes et Ifs pour la coloration), on pourrait réutiliser la moitié de ces valeurs pour calculer le dessin après un zoom de coefficient 2 (soit 3/4 pour un zoom de coefficient 1.5). Théoriquement, cet algorithme permettrait de gagner près de 50 % de vitesse pour un zoom de coefficient 2, et donc 75% pour un zoom de coefficient 1.5.

Déplacement et zoom souris

Développer un environnement beaucoup plus lié à la souris aurait été préférable pour l'utilisateur. Celui-ci aurait pu zoomé avec un clic droit, dézoomé avec un clic gauche, se déplacer en « drag and drop », ce qui est je pense, aurait été plus plaisant pour l'utilisateur. Nous n'avons pas ajouté cela pour deux raisons. Tout d'abord cela aurait pris un certain temps à développer, temps que nous n'avions pas forcément. En effet, si nous commencions à réaliser une gestion des évènements à la souris, et que nous souhaitions la réaliser correctement, il aurait fallu insérer des boutons sur l'écran pour les différents événements, choses que nous n'avions pas le temps de faire. Ensuite, cette option n'est pas compatible avec l'un des objectifs secondaires que nous avions réalisé, le dessin de la fractale de Julia avec mise à jour en temps réel suivant la position du curseur de la souris. Il aurait alors fallu gérer les évènements de deux manières différentes suivant la fractale, ce qui est inconfortable pour l'utilisateur.

Conclusion

Alexandre Ledit

Le jour où nous avons reçu le sujet de ce projet, je savais qu'il serait long mais très intéressant. Je connaissais de par mes lectures déjà quelques information importantes sur les fractales. Les modèles de Julia et Mandelbrot m'étaient familier, en revanche les autres modèles m'étaient complètement inconnus, leur compréhension n'a d'ailleurs pas été évidente. Même à la fin de ce projet je me demande encore comment par exemple une fractales IFS construite à partir de lois aussi aléatoires peuvent modéliser des formes les plus complexes et précise possible. Les fractales sont pour moi une magnifique application artistique des mathématiques et de l'algorithmique.

Je sais que l'organisation de notre projet n'est apparue que relativement tard. En effet il été demandé au début de définir des organigrammes de fonctionnement du projet, mais ne sachant pas du tout comment partir et comment notre projet allé fonctionner, nous n'avons pas produit précisément ces schémas.

La recherche des méthodes de dessin de ces fractales fut tout aussi intéressante, rechercher des heures durant des informations parfois fausses ou contradictoires m'ont permis de devenir beaucoup plus méfiant quant à l'information qui circule. De même la partie optimisation du code est une partie d'un projet que nous ne connaissions pas mais qui est très intéressante et demande beaucoup de réflexion. Nous avons vu et pensé vers la fin de ce projet à des optimisations que nous n'avons pas pu mettre en place par manque de temps ce qui est un peu dommage.

Antoine Souplet

Un projet intéressant

Tout d'abord, c'était un projet qui allie la programmation à du visuel. C'est-à-dire que le rendu de notre programmation, pouvait se tester et s'apprécier presque immédiatement sur notre fenêtre SDL. Il est beaucoup plus facile et agréable de programmer, lorsque le rendu visuel du projet est important et progresse parallèlement au code.

Ensuite, j'ai bien apprécié le mode de fonctionnement très « libre », qui nous place dans une situation ou nous devons rechercher des informations par nous même, et progresser grâce aux erreurs commises. D'autant plus qu'il n'y avait pas tant d'informations que cela sur internet, notamment pour flamme, où nous avancions légèrement dans l'inconnu. Ensuite, j'ai apprécié le fait que les groupes ne soient pas de taille trop important, comme lors du projet Productel. En effet, plus le nombre de personne associé à un projet est important, plus l'implication des membres est faible (je le reconnais pour ma part également). Ainsi, le fait de travailler en duo sur un projet, nous permet de nous l'approprié de telle sorte que nous sommes énormément impliqué dedans, et d'autant plus sujet à l'envie de le réussir.

Enfin j'ai bien aimé ce projet car il n'admettait pas réellement de fin, il était toujours possible de l'améliorer, aussi bien au niveau de la performance et rapidité du calcul, qu'au niveau de la coloration ou d'autres fonctionnalités. Le projet aurait pu se terminer dans 3 mois, nous aurions encore des choses à améliorer ; zoomer toujours plus rapidement et à l'infini, créer une vraie interface graphique, faire en sorte que le programme permettent d'ouvrir plusieurs fenêtre de dessin de différentes fractales sans quitter le programme, permettre à l'utilisateur de régler lui-même sa coloration avec des curseurs de couleurs RGB, gérer un évènement sans attendre que le précèdent se termine… Les améliorations sont infinies, et les idées ne manquent pas, et c'est ce type de projet que j'aime.

Un projet bénéfique

Ce projet m'a été bénéfique pour, je pense, tous les projets à venir. Tout d'abord au sujet de la planification du projet, qui je pense pour ce projet, n'a pas été assez précise ni réellement mise à jour. Néanmoins, ce projet était pour ma part très particulier dans le sens ou il m'était très difficile de m'imaginer à quoi pouvait correspondre le fait de dessiner des fractales sans m'être réellement plongé dans le code. De ce fait la planification s'est avérée très complexe.

Ensuite, d'un point de vue plus informatique, le projet m'a surtout permit d'étendre fortement mon niveau de programmation en langage C, par rapport à celui acquis en CPI1, époque à laquelle je ne maitriser ni les pointeurs, ni l'allocation dynamique, et encore moins les notions de « fuites mémoires ». De plus, m'a fait prendre conscience de l'intérêt de certaines règles de programmations, même si celle-ci peuvent paraîtres rébarbatives lors des premières semaines d'application.

Enfin, sur un plan plus général, ce projet m'a apprit énormément de chose sur les fractales, notion sur laquelle je n'avais quasiment aucune connaissance.

Sources

Magazines Sciences & Vie

www.wikipedia.org

www.futura-sciences.com