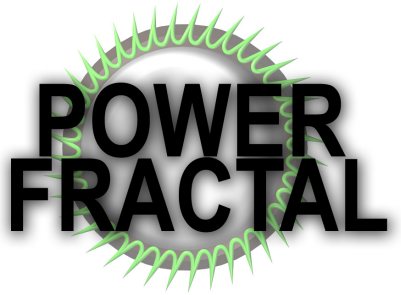
ANNEXE B

Documentation Technique



G.Burri & A.Crivelli
[EIA]

Table des matières

1. Informations générales

1.1 Matériel nécessaire à l'exécution du logiciel

Power Fractal tourne sur plateforme Windows 4. x ainsi que Nt 4.x et Nt 5.X.

Comme le calcul de fractal demande beaucoup de calcul il est préférable de posséder un ordinateur assez puissant, nous recommandons au minimum un processeur cadencé à 233Mhz et pour un confort d'utilisation optimal un processeur cadencé à 400Mhz ou plus.

En ce qui concerne la mémoire vive, 32 Mo suffiront amplement (cela dépend de votre system d'exploitation, par exemple les systems basés sur le noyau NT demande une plus grande ressource au niveau de la mémoire vive).

1.2 Matériel nécessaire à la compilation du logiciel

Ce logiciel a était compilé avec Object Ada 7.2. De plus la bibliothèque spider (libre de droit) est nécessaire.

1.3 Où peut-on se procurer la source ?

Pour vous procurer le logiciel ainsi que les sources vous pouvez envoyer un e-mail à :

Greg.burri@net2000.ch ou Powerkiki@urbanet.ch

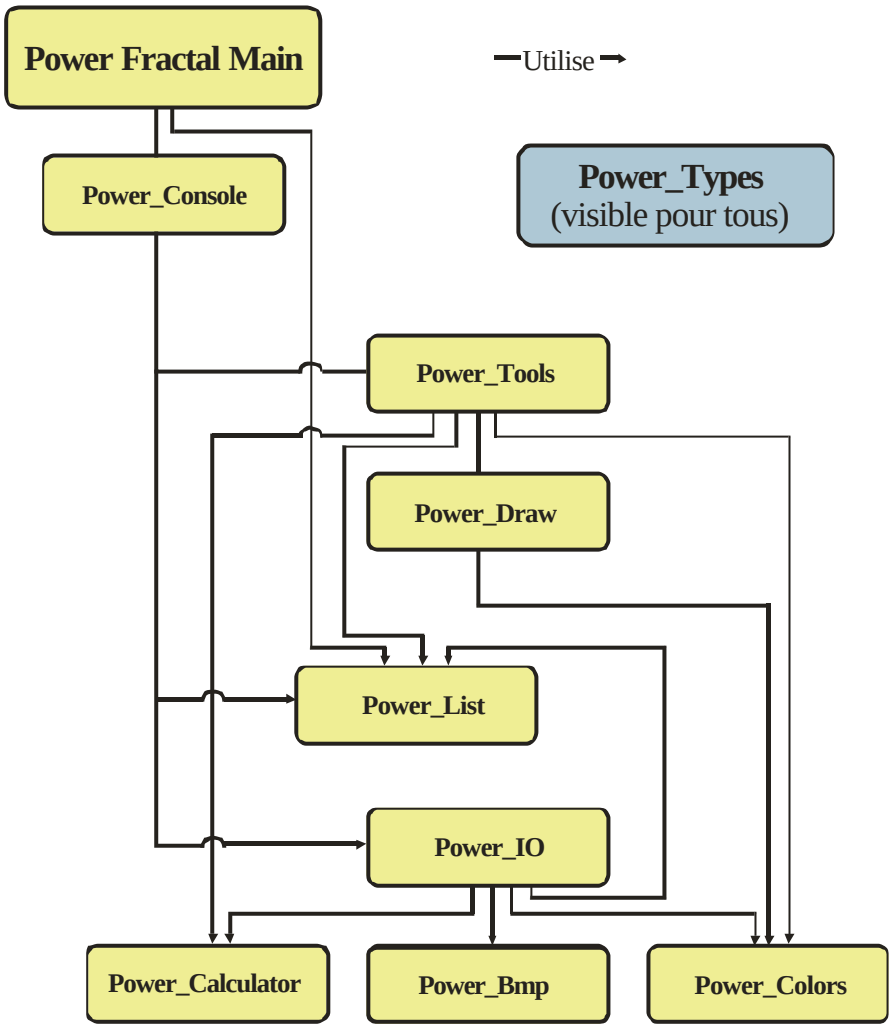
Ou plus simplement en allant sur le site officiel :

http://pifou.servehttp.com/powerfractal

Remarque : ce logiciel est sous licence GPL (General public license).
Pour connaître les termes de la licence :
- voir le document GPL
- site : http://www.gnu.org

1.4 Structure générale du programme

Voici la structure générale de Power_Fractal :



Brève explication des unités :

Power_Types :
Contient les types, les constantes et les variables globales.
Exemple :
- Le tampon écran (var)
- La définition d'une fractal

Power_Fractal_Main :
C'est le programme principal, c'est par lui que commence le programme. Il ne fait rien d'autre que d'initialiser la fenêtre graphique, de créer une liste de fractals et de lancer la console.

Power_Console :
Gère l'interface utilisateur. Permet d'interpréter les différentes commandes tapées par celui-ci.

Power_Tools :
Met à disposition les outils pour dessiner la fractal dans la fenêtre graphique, pour 'naviguer' dans la fractal comme la fonction zoom et la fonction pour recentrer la fractal.
Contient également la procédure de choix à la souris d'une fractal de la liste.

Power_Liste :
Contient toutes les opérations de gestion de la liste de fractals : par exemple : effacer ou ajouter une fractal.

Power_IO :
Permet de faire des entrées/sorties dans des fichiers :
- Rendre une fractal en bmp.
- Sauver/charger une fractal
- Sauver/charger une liste de fractals.

Power_Draw :
Outils de base pour dessiner sur la fenêtre graphique :
- Dessiner un cadre.
- Dessiner/effacer une croix.
- Dessiner une partie du tampon d'affichage.

Power_Calculator :
Contient tout ce qui touche au calcul : pour calculer les fractals, l'antialiasing ou l'angle.

Power_bmp :
Ne contient qu'une procédure permettant d'écrire un fichier bmp.

Power_Colors :
Contient les outils de calculs de couleur.
Sert à calculer un dégradé a partir de plusieurs couleurs et à l'appliquer à la fractal.

2. Liste des procédures, types, variables et constantes (headers)

2.1 Power_Types

Constantes

Nom	*Type*	*Initialisation*
Hauteur_Ecran		399
Largeur_Ecran		639
Rayon_Int_Croix		2
Rayon_Ext_Croix		6
Hauteur_Degrade		5
Facteur_Zin		4.0
Facteur_Zout		4.0
Zoom_Min		0.1
Nb_Iteration_Min		6
Longueur_Max		50
Nb_Couleur_Max		10
Prompt	String	"> "
Couleur_Cadre	Spider.Draw.Tcolor	(255 , 255 , 200)
Couleur_Numeros	Spider.Draw.Tcolor	(255 , 255 , 200)
Echape	Character	Ada.Characters.Latin_1.Esc
Vide	Character	Ada.Characters.Latin_1.Nul
Espace	Character	Ada.Characters.Latin_1.Space
Fractal_Initial_Julia	Cara_Fractal	(Julia , 4 , 50 , 4.0 , 1.0 / 4.0 , (0.0 , 0.0) , 0.0 , ((255 , 50 , 50) , (200 , 254 , 120) , (123 , 78 , 100) , (0 , 23 , 100)) , False , False , 0.577 , 0.468)
Fractal_Initial_Mandel	Cara_Fractal	(Mandelbrot , 4 , 50 , 4.0 , 1.0 / 4.0 , (0.6 , 0.0) , 0.0 , ((100 , 100 , 255) , (120 , 230 , 255) , (123 , 78 , 100) , (0 , 23 , 100)) , False , False)

Variables

Nom	*Type*	*Initialisation*
Matrice_Iteration_Global	T_Matrice_Iteration_2	
Matrice_Tampon_Ecran	T_Matrice_Tampon(0 .. Largeur_Ecran , 0 .. Hauteur_Ecran)	

Types

Nom	*Déclaration*
Byte	mod 256
T_Matrice_Tampon	array (Natural range <> , Natural range <>) of T_Couleur
T_Ensemble	(Julia , Mandelbrot)
T_Tab_Couleur	array (Natural range <>) of T_Couleur
T_Matrice_Iteration	array (Integer range <> , Integer range <>) of Natural
T_Lien	access T_Fractal

Sous types

Nom	*Déclaration*
T_Nb_Couleur	Integer range 2 .. Nb_Couleur_Max

T_Couleur *(article)*

Champ	*Type*	*par défaut*
R	Byte	
G	Byte	
B	Byte	

T_Centre *(article)*

Champ	*Type*	*par défaut*
X	Long_Float	0.6
Y	Long_Float	0.0

Cara_Fractal *(article)*		
Discriminant	***Type***	***par défaut***
Ensemble	T_Ensemble	Mandelbrot
Nb_Couleur	T_Nb_Couleur	4
Champ	***Type***	***par défaut***
NB_Iteration_Max	Positive	50
C_Diverge_Limite	Long_Float	4.0
Zoom	Long_Float	1.0 / 4.0
Centre	T_Centre	(0.6 , 0.0) Angle : Long_Float 0.0
Couleur	T_Tab_Couleur(1..Nb_Couleur)	((255 , 0 , 0) (0 , 255 , 0) (0 , 0 , 255) (255 , 255 , 0)
Antialiasing	Boolean	False
Dessine_Degrade	Boolean	False
Partie variante		
Ensemble = Julia		
Cx	Long_Float	0.577
Cy	Long_Float	0.468
Ensemble = Mandelbrot		
null		

T_Matrice_Iteration_2 *(article)*		
Discriminant	***Type***	***par défaut***
Antialiasing	Boolean	False
Champ	***Type***	***par défaut***
Partie variante		
Antialiasing = False		
Matrice_Iteration	T_Matrice_Iteration(0..Largeur_Ecran,0..Hauteur_Ecran)	
Antialiasing = True		
Matrice_Iteration_Anti	T_Matrice_Iteration(0..Largeur_Ecran*2+1,0..Hauteur_Ecran*2+1)	

T_Fractal *(article)*		
Champ	***Type***	***par défaut***
Fractal	Cara_Fractal	
Suiv	T_Lien	

T_Liste_Fractals *(article)*		
Champ	***Type***	***par défaut***
Tete	T_Lien	null
Nb_Fractals	Natural	0

2.2 Power_Console

Clauses de contextes	
with	Power_Types
with	Power_List

Clauses USE	
use	Power_Types
use	Power_List

Console *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE_SORTIE	Liste	T_Liste_Fractals		
ENTREE	Prompt	String		

2.3 Power_Tools

Clauses de contextes	
with	Power_Types

Clauses USE	
use	Power_Types

Dessiner_Fractal *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Fractal	Cara_Fractal		
ENTREE	X	Natural	0	
ENTREE	Y	Natural	0	

Rafraichir_Couleur *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Fractal	Cara_Fractal		

Zoom_Souris *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE_SORTIE	Fractal	Cara_Fractal		

Centrer *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE_SORTIE	Fractal	Cara_Fractal		

Choix_Mosaic *(fonction)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Liste_Fractals	T_Liste_Fractals		
RETOUR		Natural		

2.4 Power_List

Clauses de contextes	
with	Power_Types

Clauses USE	
use	Power_Types

Exceptions	
Liste_Une_Fractal	
Fractal_Inexistante	

2.5 Power_IO

Clauses de contextes	
with	Power_Types
use type	Power_Types.Cara_Fractal

Exceptions	
Erreur_Fichier	

Enregistrer_Fractal *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Nom_Fichier	String		
ENTREE	Fractal	Power_Types.Cara_Fractal		

Charger_Fractal *(fonction)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Nom_Fichier	String		
RETOUR		Power_Types.Cara_Fractal		

Enregistrer_Liste *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Nom_Fichier	String		
ENTREE	Liste	Power_Types.T_Liste_Fractals		

Charger_Liste *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Nom_Fichier	String		
ENTREE_SORTIE	Liste	Power_Types.T_Liste_Fractals		

Rendre_Bmp *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Fractal	Power_Types.Cara_Fractal		
ENTREE	Nom_Fichier	String		
ENTREE	Largeur_Zone	Natural	1024	
ENTREE	Hauteur_Zone	Natural	768	

2.6 Power_Draw

Clauses de contextes	
with	Spider.Draw

Boite *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	X1	Natural		
ENTREE	Y1	Natural		
ENTREE	X2	Natural		
ENTREE	Y2	Natural		

Ligne_Matrice_Hori *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	X1	Natural		
ENTREE	X2	Natural		
ENTREE	Y	Natural		

Ligne_Matrice_Vert *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Y1	Natural		
ENTREE	Y2	Natural		
ENTREE	X	Natural		

Dessin_Croix ***(procédure)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	X	Natural		
ENTREE	Y	Natural		

Efface_Croix ***(procédure)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	X	Natural		
ENTREE	Y	Natural		

2.7 Power_Calculator

Clauses de contextes	
with	Power_Types

Clauses USE	
use	Power_Types

Calcul_Antialiasing ***(fonction)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Matrice_Antialiasing	T_Matrice_Tampon		
RETOUR		T_Matrice_Tampon		

Change_Angle ***(procédure)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE_SORTIE	A	Long_Float		
ENTREE_SORTIE	B	Long_Float		

Mandel_Gen ***(fonction)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Largeur_Zone	Natural		
ENTREE	Hauteur_Zone	Natural		
RETOUR		T_Matrice_Iteration		

Julia_Gen *(fonction)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Largeur_Zone	Natural		
ENTREE	Hauteur_Zone	Natural		
RETOUR		T_Matrice_Iteration		

2.8 Power_Bmp

Clauses de contextes	
with	Power_Types

Clauses USE	
use	Power_Types

Ecrire_Bmp *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Matrice_Image	T_Matrice_Tampon		
ENTREE	Nom_Fichier	String		

2.9 Power_Colors

Clauses de contextes	
with	Spider.Draw
with	Power_Types

Inverse_Couleur *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE_SORTIE	Couleur	Spider.Draw.Tcolor		

Creer_Degrade *(procédure)*				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Fractal	Power_Types.Cara_Fractal		
SORTIE	Degrade	Power_Types.T_Tab_Couleur		
ENTREE	Longueur	Integer		

Affiche_Degrade ***(procédure)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Fractal	Power_Types.Cara_Fractal		

Conversion_Couleur ***(fonction)***				
Paramètres	***Nom***	***Type***	***par défaut***	
ENTREE	Matrice	Power_Types.T_Matrice_Iteration		
ENTREE	Degrade	Power_Types.T_Tab_Couleur		
RETOUR		Power_Types.T_Matrice_Tampon		

3. Description des fichiers

2.1 Liste des fichiers

- **PowerFractal.ada**
- **Power_Bmp.ads**
- **Power_Bmp.adb**
- **Power_Calculator.ads**
- **Power_Calculator.adb**
- **Power_Colors.ads**
- **Power_Colors.adb**
- **Power_Console.ads**
- **Power_Console.adb**
- **Power_Draw.ads**
- **Power_Draw.adb**
- **Power_IO.ads**
- **Power_IO.adb**
- **Power_List.ads**
- **Power_List.adb**
- **Power_Tools.ads**
- **Power_Tools.adb**
- **Power_Types.ads**

4. Description générale

4.1 Les principales structures de données

Toutes les structures décrites ici ce trouve dans l'unité Power_List.

A) **La structure d'une fractal :**

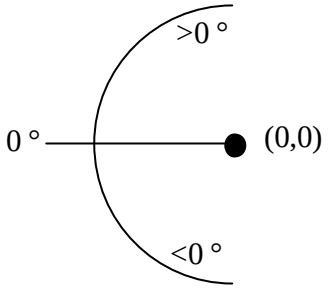
```
type Cara_Fractal (Ensemble : T_Ensemble := Mandelbrot; Nb_Couleur :
T_Nb_Couleur := 4) is
    record
        NB_Iteration_Max : Positive   := 50;
        C_Diverge_Limite : Long_Float := 4.0
        Zoom    : Long_Float := 1.0 / 4.0;
        Centre  : T_Centre   := (-0.6, 0.0);
        Angle   : Long_Float := 0.0;
       Couleur : T_Tab_Couleur(1 .. Nb_Couleur) := ((255,0,0), (0,255,0),
                                    (0,0,255), (255,255,0) );
        Antialiasing : Boolean := False;
        Dessine_Degrade : Boolean := False;
    case Ensemble is
        when Julia =>
            Cx : Long_Float := -0.577;
            Cy : Long_Float := 0.468;
        when Mandelbrot =>
            null;
        end case;
    end record;
```

Explication:

Le type Cara_Fractal (caractéristique d'une fractal) donne tout les renseignement pour pouvoir tracer une fractal à l'écran :

- Le type de fractal : Mandelbrot ou Julia *(Ensemble)*.
- Le nombre d'itérations à calculer *(NB_Iteration_Max)*.
- La constante : limite de divergence (voir doc : Que est ce qu'une fractal). *(C_Diverge_Limite)*.
- Le zoom *(Zoom)*.
- Le point de la fractal qui se situe au centre de l'écran (Centre).

- L'angle de rotation de la fractal : (absolu)



- Les couleurs de la fractals qui sont représentées par un tableau de longueur variante (max 10 couleurs)

- Une information concernant l'antialiasing, si il est appliqué ou pas (Antialiasing)

- Une information pour savoir si le dégradé doit être affiché ou pas. Dessine_Degrade

- Des informations concernant la constante 'c' (voir doc : Que est ce qu'une fractal), elle n'est utilisé uniquement dans le calcul de l'ensemble de julia, c'est pourquoi un article à parties variantes est utilisé.

B) **La matrice contenant la fractal calculé :**

```
type T_Matrice_Iteration_2 (Antialiasing : Boolean := False) is
    record
    case Antialiasing is
        when False =>
            Matrice_Iteration : T_Matrice_Iteration (0..Largeur_Ecran,
            0..Hauteur_Ecran);
        when True =>
            Matrice_Iteration_Anti : T_Matrice_Iteration
            (0..Largeur_Ecran * 2 + 1, 0..Hauteur_Ecran * 2 + 1);
        end case;
    end record;
```

Explication :

Type utilisé dans la variable global *'Matrice_Iteration_Global'* qui sert à garder en mémoire le calcul de la fractal courante. Cela sert lorsque l'on change le dégradé de couleur qui va être appliqué directement à cette matrice sans passer par la phase de calcul de la fractal.
Le type contient en fait deux matrices, une quadruple de l'autre pour lui appliquer le calcul de l'antialiasing et l'autre simple (taille de la fenêtre).

4.2 L'algorithme de calcul de fractal

L'explication ci-dessous est fait pour l'ensemble de Mandelbrot, le calcul de l'ensemble de Julia et quasiment identique à une constante près.

La formule générale est :

$Z_{(n+1)} = Z_{(n)}^2 + Z_{(0)}$ $Z_{(0)}$ est la valeur initiale

où Z est un nombre complexe qui peut être représenté par un point sur un plan, et justement le plan sera la fenêtre graphique et les points les pixels.

Si l'on analyse vers quelle valeur tend cette fonction pour chacun des points du plan. On constate que pour beaucoup de point la fonction diverge plus ou moins rapidement et au contraire pour certains points la fonction tend vers une valeur. C'est justement ces point, qui ne diverge pas, qui forme l'ensemble de mandelbrot.

Le problème maintenant et de savoir si la fonction diverge en un point donné du plan. Pour cela il suffit de vérifier que le module de Z reste inférieur ou égale à une valeur de référence qui est 2 (valeur d'échappement).

Rappel : Le module de $Z = a + b \cdot j$ est $\sqrt{a^2 + b^2}$

Du point de vu de l'algorithme :
Nous allons donc effectuer une boucle, et à chaque passage calculer le Z suivant. Il faut aussi tester le module (voir ci-dessus) pour savoir si il dépasse 4, nous le feront par une condition d'arrêt.

Le problème maintenant c'est que si en un point (pixel dans notre cas) la fonction ne diverge pas alors le module restera en dessous de 2 et nous ne sortirons jamais de la boucle, il faut donc fixer une limite de calcul, pour cela nous utiliseront une variable que nous incrémenteront de 1 à chaque passage.

Si on se fixe par exemple 200 comme limite du nombre d'itération alors, lorsque cette valeur sera atteinte il suffira de sortir de la boucle, dans ce cas on ne pourra pas déterminer si la fonction diverge ou ne diverge pas, mais au moins penser que c'est un des point qui diverge le moins.

Un autre avantage de cette variable et de donner une pseudo valeur de divergence lorsque la boucle se termine par la condition de test du module de Z. En effet, si la fonction diverge rapidement alors la module atteindra rapidement la valeur 2 et la variable comptant le nombre d'itération n'aura pas beaucoup était incrémentée. Si au contraire Z mets plus de 'temps' pour atteindre la valeur d'échappement alors la variable aura était 'plus' incrémentée.

Pour résumer si la valeur de la variable est petite, la fonction diverge rapidement. Si elle est grande alors la fonction diverge plus lentement. Et si elle atteint la valeur limite d'itération (celle que l'on s'est fixé) alors on peut présumer que le point ne diverge pas ou alors très peut.

Pour augmenter la précision de calcul, il suffit de mettre la limite plus haut. Mais il faut être raisonnable, sur un écran d'ordinateur la précision de l'affichage est limité par les pixels, il est donc inutile, passé une certaine limite, d'effectuer des calculs supplémentaires.

Après avoir effectué tous les calculs, nous obtenons une matrice contenant le nombre d'itération pour chaque point de l'écran. Pour afficher cette matrice à l'écran et obtenir de surcroît un effet visuel intéressant nous allons convertir ces 'nombre d'itération' en couleurs. Pour que l'on voit bien la direction des champs de divergence, un dégradé sera appliqué a la matrice.

Exemple :
Pour une matrice dont les valeurs varient entre 1 et 100 (itération(s)) on va superposer un dégradé de couleur composé de 100 étapes, par exemple du jaune au rouge, la couleur 1 étant le jaune et la couleur 100 étant le rouge. Donc ce dégradé comporte 100 couleurs différentes et à chaque itération une couleur correspond.
Nous allons donc créer une nouvelle matrice de même dimension que la première, cela va de soit, et la remplir de valeurs de couleur en fonction de la première matrice.

Voici le pseudo code de l'algorithme de calcul de la première matrice (matrice d'itération) :

M est une matrice d'entiers positif de dimension X Y (on prendra la taille de la zone de dessin de la fenêtre graphique)

N est le nombre d'itération courant
N_{max} est le nombre d'itération limite

Z est le nombre complexe
$Z_{(0)}$ est le nombre complexe de départ

Pour chaque point x y de la matrice

$N = 0$
$Z_{(0)} = x + y \cdot i$ #nombre complexe
$Z = Z_{(0)}$ #pour ne pas effacer $Z_{(0)}$, on en a besoin !

Boucle
$Z = Z^2 + Z_{(0)}$
N++
Sort si N = N_{max} ou si $\sqrt{x^2 + y^2} > 2$ (module de *Z*)

M(x, y) = N

4.3 Comment passer des coordonnées d'un point de l'écran aux coordonné réelles de la fractal ?

Pour faire tous nos calculs sur les fractals nous avons besoin de faire un lien entre les coordonnées des pixels de l'écran et les coordonnées sur la fractal.

Voici la formule qui a était établie pour la dimension X:

$$a = \frac{\frac{X}{L} - 0.5}{Zoom} + CentreX$$

Variables :

a	:	Valeur réel du complexe
X	:	Valeur de la dimension X sur l'écran
L	:	Longueur de l'écran (dimension X)
Zoom	:	Facteur de zoom
CentreX	:	La valeur en x du point de la fractal que l'on veut au centre de l'écran

Explication :

X/L donne une valeur comprise entre 0 et 1 que nous allons décaler de 0.5 pour avoir la coordonnée 0 au centre de l'écran
Puis nous divisons par le *zoom*.
En enfin on décale la fractal pour avoir la coordonnée *CentreX* au centre de l'écran

Voici la formule qui a était établie pour la dimension Y:

$$b = \frac{-2 \cdot Y + H}{2 \cdot L \cdot Zoom} + CentreY$$

Variables :

b	:	Valeur imaginaire du complexe
Y	:	Valeur de la dimension Y sur l'écran
L	:	Longueur de l'écran (dimension X)
H	:	Hauteur de l'écran
Zoom	:	Facteur de zoom
CentreY	:	La valeur en y du point de la fractal que l'on veut au centre de l'écran

Pour mieux comprendre cette formule nous la développons comme ceci :

$$b = \left(\frac{-Y}{L} + \frac{H}{2 \cdot L} \right) \cdot \frac{1}{Zoom} + CentreY$$

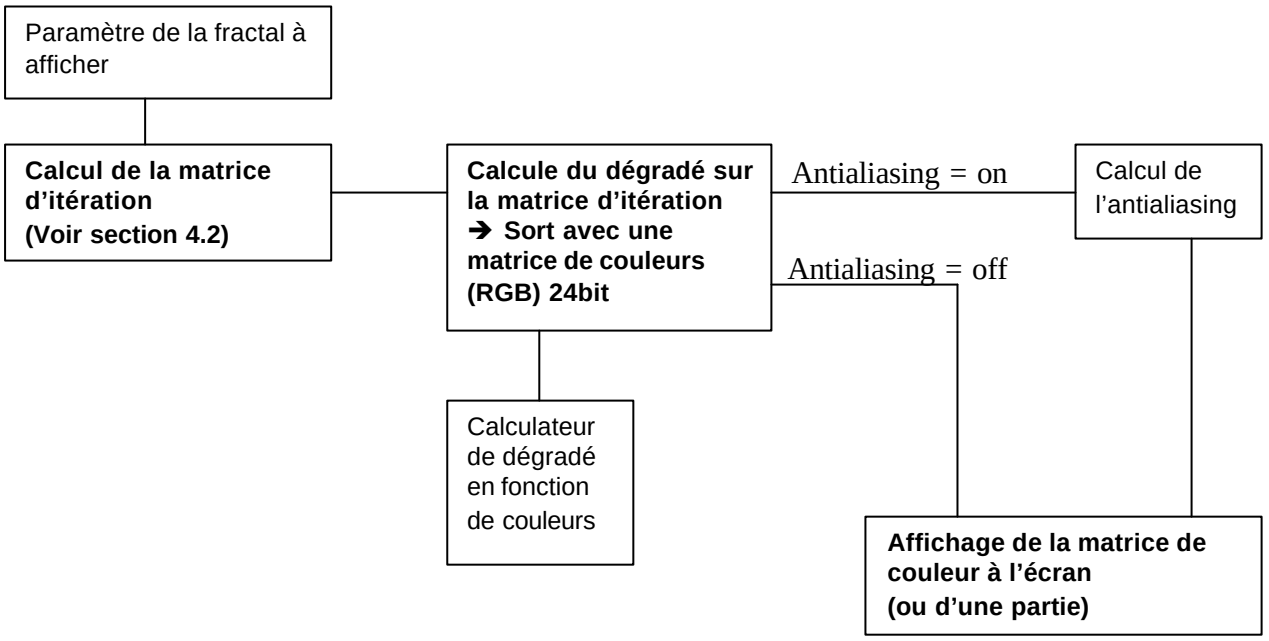
-Y/L représente la valeur ramené sur l'axe X, le moins sert a faire une homothétie d'axe y de la fractal.

*H/(2*L)* est le rapport en la hauteur et la largeur, il faut garder les mêmes proportions. Le 2 sert à mettre la coordonnée 0 au centre de l'écran.

Puis on applique le *Zoom* et on décale à l'aide de *CentreY*.

4.4 Les étapes de calculs de la fractal finale.

Voici un schéma bloc montrant les différentes étapes de calcules :

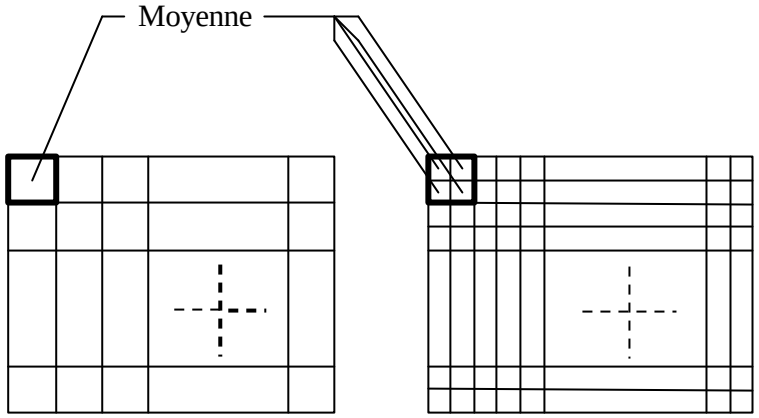


Que est ce que l'antialiasing et comment se calcul-t-il ?

L'antialiasing veut dire en français 'réductions des défauts visuels'.
Dans notre cas c'est un lissage qui est effectué, la matrice de sortie (tampon d'affichage) sera de même taille et de même nature que sans antialiasing mais elle sera porteuse d'une quantité d'information plus élevée.

L'antialiasing se calcul d'une façon très simple :
On calcul en premier lieu une matrice quatre fois plus grande (que la taille normal) se qui demande, bien entendu, quatre fois plus de temps de calcul mais aussi quatre fois plus de mémoire !
Puis on va lui appliquer la transformation en fonction des couleurs pour obtenir la matrice de couleurs (quatre fois plus grande elle aussi).
Puis enfin vient le calcul de l'antialiasing :
On va réduire la taille de la matrice de quatre fois en faisant une moyenne de quatre points adjacents pour n'en former plus qu'un, ce qui nous donne à l'arrivé une matrice de la taille souhaité mais porteuse de plus d'informations.

Schéma explicatif (de droite à gauche)



Cette technique permet une qualité visuelle nettement au dessus d'un affichage sans antialiasing, elle est surtout utilisée lors de rendu final dans un fichier bmp. Il ne faut pas oublier que cette technique consomme énormément plus de ressource system !