Introduction à la Programmation

—

TP #7

—

G1: monasse*(at)*imagine.enpc.fr G2: nicolas.audebert*(at)*onera.fr
G3: alexandre.boulch*(at)*onera.fr G4: maxime.ferrera*(at)*onera.fr
G5: laurent.bulteau*(at)*u-pem.fr G6: pierre-alain.langlois*(at)*eleves.enpc.fr

1 Premiers objets et dessins de fractales

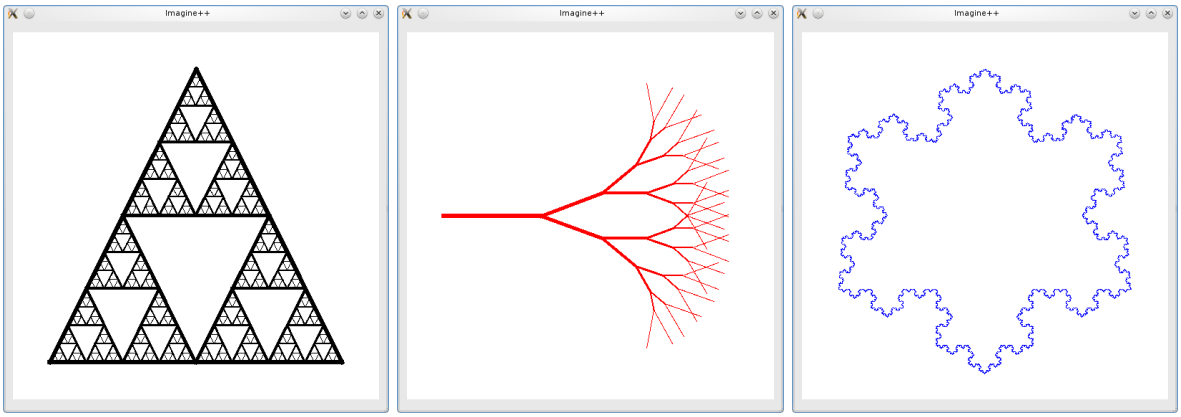

FIGURE 1 – Fractales...

Dans ce TP, nous allons nous essayer à la programmation objet. Nous allons transformer une structure vecteur en une classe et l'utiliser pour dessiner des courbes fractales (figure 1).

1.1 Le triangle de Sierpinski

1. *Récupérer le projet :*
   Télécharger le fichier `Tp7_Initial.zip` sur la page habituelle, le décompresser et lancer votre IDE. Etudier la structure `Vector` définie dans les fichiers `Vector.cpp` et `Vector.h`.
2. *Interfaçage avec Imagine++ :*
   La structure `Vector` ne comporte pas de fonction d'affichage graphique. Ajouter dans `main.cpp` des fonctions `drawLine` et `drawTriangle` prenant des `Vector` en paramètres. Il suffit de rebondir sur la fonction

   ```
   void drawLine(int x1,int y1,int x2,int y2,const Color& c,int pen_w)
   ```

   d'Imagine++. Le dernier paramètre contrôle l'épaisseur du trait.
3. *Triangle de Sierpinski :*
   C'est la figure fractale choisie par l'ENPC pour son logo. La figure ci-dessous illustre sa construction.
   Écrire une fonction récursive pour dessiner le triangle de Sierpinski. Cette fonction prendra en paramètres les trois points du triangle en cours et l'epaisseur du trait. Les trois sous-triangles seront dessinés avec un trait plus fin. **Ne pas oublier la condition d'arrêt de la récursion !**
   Utiliser cette fonction dans le main en lui fournissant un triangle initial d'épaisseur 6.

1.2 Une classe plutôt qu'une structure

4. *Classe vecteur :*
   Transformer la structure `Vector` en une classe. Y incorporer toutes les fonctions et les opérateurs. Passer en public le strict nécessaire. Faire les modifications nécessaires dans `main.cpp`.

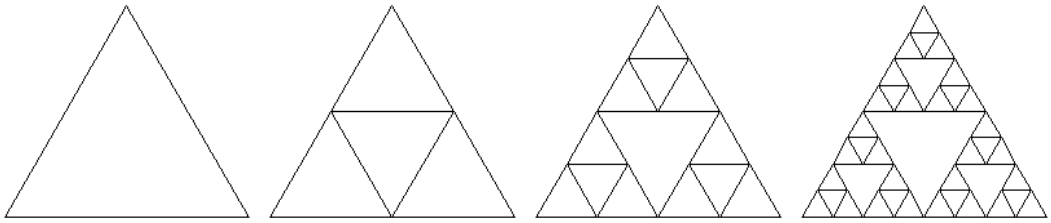

FIGURE 2 – Construction du triangle de Sierpinski.

5. *Accesseurs pour les membres :*
   Rajouter des accesseurs en lecture et en écriture pour les membres, et les utiliser systématiquement dans le programme principal. L'idée est de cacher aux utilisateurs de la classe `Vector` les détails de son implémentation.
6. *Dessin récursif d'un arbre :*
   Nous allons maintenant dessiner un arbre. Pour cela il faut partir d'un tronc et remplacer la deuxième moitié de chaque branche par deux branches de même longueur formant un angle de 20 degrés avec la branche mère. La figure ci-dessous illustre le résultat obtenu pour différentes profondeurs de récursion.
   Écrire une fonction récursive pour dessiner une telle courbe. Vous aurez besoin de la fonction `Rotate`

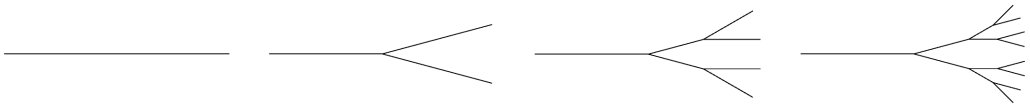

FIGURE 3 – Construction de l'arbre.

de la classe `Vector`.

1.3 Changer d'implémentation

7. *Deuxième implémentation :*
   Modifier l'implémentation de la classe `Vector` en remplaçant les membres double x,y; par un tableau double coord[2];. Quelles sont les modifications à apporter dans `main.cpp` ?
8. *Vecteurs de dimension supérieure :*
   L'avantage de cette dernière implémentation est qu'elle se généralise aisément à des vecteurs de dimension supérieure. Placer une constante globale `DIM` égale à 2 au début de `Vector.h` et rendre la classe `Vector` indépendante de la dimension.
   NB : la fonction `Rotate` et les accesseurs que nous avons écrits ne se généralisent pas directement aux dimensions supérieures. Les laisser tels quels pour l'instant...

1.4 Le flocon de neige

9. *Courbe de Koch :*
   Cette courbe fractale s'obtient en partant d'un segment et en remplaçant le deuxième tiers de chaque segment par deux segments formant la pointe d'un triangle équilatéral.
   Écrire une fonction récursive pour dessiner une courbe de Koch.

FIGURE 4 – Construction de la courbe de Koch.

10. *Flocon de neige :*
    Il s'obtient en construisant une courbe de Koch à partir de chacun des côtés d'un triangle équilatéral.