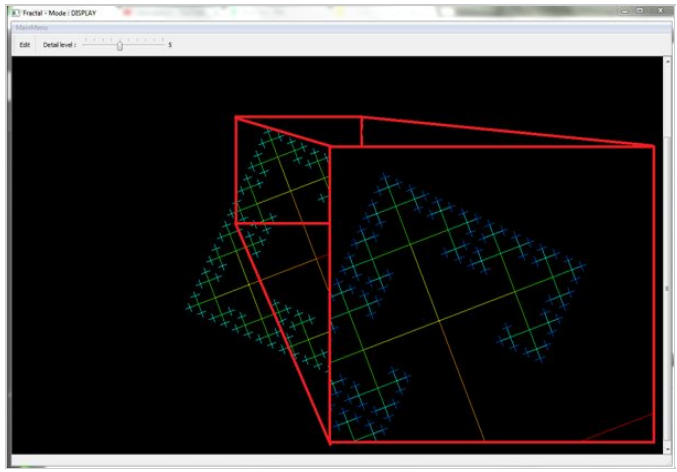
Luy Karim
Lovis Thomas
Rodrigues L. Daniel

TP1 - Singleton & Composite



Le code Qt d'écrit dans ce document génère et affiche une fractale. Celui-ci utilise les deux patrons de conceptions *« Singleton »* et *« Composite »*.
Le programme est constitué d'une classe *« Fractal »* implémenté avec le patron *« Singleton »* et qui contient une fractale construite en suivant le patron *« Composite »*.

Singleton

La classe *« Fractal »* ci-dessus se base sur le patron de conception *« Singleton »*. En effet, on voit que son constructeur est privé et qu'elle utilise une méthode publique *« getInstance () »*.

```
class Fractal
{
public:
    enum ItemAttribut{ItemDepth,ItemIsLeaf};
    ~Fractal();
    static Fractal* getInstance(QLineF segment, int profondeur);
    void static deleteAllInstance();
    //DP composite
    void drawFractal(QGraphicsScene *& scene);
private:
    Fractal(QLineF segment, int profondeur);
```

Cette méthode *« getInstance () »* renvoie une instance de cette classe si elle existe sinon elle est construite comme démontré ci-dessous.

```
Fractal* Fractal::getInstance(QLineF segment, int profondeur)
{
    if(listFractal.length() < N_MAX )
    {
        listFractal.append(new Fractal(segment,profondeur));
        return listFractal.last();
    }
    else
        return 0;
}
```

On remarque que dans *« getInstance () »* on ne se contente pas de créer une instance ou de renvoyer celle-ci si elle existe déjà. La classe *« Fractal »* peut en réalité contenir plusieurs instances d'elle-même donc d'afficher plusieurs fractales.
Le nombre d'instance est limité par la constante *« N_MAX »*. Dans le code fournit avec ce rapport *« N_MAX »* vaut 3 (donc 3 instances possibles), mais si *« N_MAX »* valait 1, la classe *« Fractal »* respecterait pleinement le patron *« Singleton »*.

Luy Karim
Lovis Thomas
Rodrigues L. Daniel

Voici l'appel de la méthode *« getInstance () »* dans le widget principal renvoyant l'instance d'une fractale.

```
if( (fractal = Fractal::getInstance(segInitial,deepness)) ){
    fractal->drawFractal(scene);
    editionLines.append(segInitial);
}
```

Composite

Dans notre exemple, pour le patron de conception *« Composite »*, une feuille correspond à une ligne et un composite à une croix de ligne.
Les lignes correspondent à la classe *« LeafFractal »* et les croix à la classe *« CompositeFractal »*.
Tous deux héritent de la classe *« AbstractFractal »*.

```
class LeafFractal : public AbstractFractal
```

```
class CompositeFractal : public AbstractFractal
{
private:
    QList<AbstractFractal *> children;
```

Schéma de classes

Voici le schéma de classe du programme.