Rapport de Projet INFO0402

Valentin MAROT, Théo GODEREAUX

2ème année de licence informatique

URCA

Bibliothèque statique C++ GrawEditor :

Instanciation du canevas :

La bibliothèque permet la création d'un canevas appelé GrawEditor.

```
GrawEditor& editor = GrawEditor::GetEditor();
```

L'utilisation du modèle singleton permet de limiter l'instanciation de l'éditeur et retourne une référence vers l'éditeur existant s'il y en avait déjà un.

Création de forme :

Utilisation de la fonction template GetNew() :

Segment :

Sans paramètres :

```
editor.GetNew<Segment>();
```

Avec paramètres :

```
editor.GetNew<Segment>(Point(0,0),Point(1,1));
editor.GetNew<Segment>(Point(0,0),Point(1,1),Color::blue,2);
```

Rectangle :

Sans paramètres :

```
editor.GetNew<Rectangle>();
```

Avec paramètres :

```
editor.GetNew<Rectangle>(Point(0,0),2.f,2.f,Color::red);
```

Carré :

Sans paramètres :

```
editor.GetNew<Square>();
```

Avec paramètres :

```
editor.GetNew<Square>(Point(0,0),2.f,Color::green);
```

Triangle :

Sans paramètres :

```
editor.GetNew<Triangle>();
```

Avec paramètres :

```
editor.GetNew<Triangle>(Point(10,10),3.f,Color::green);
initializer_list<Point> l={Point(0,0),Point(1,1),Point(2,2)};
editor.GetNew<Triangle>(l,Color::yellow);
```

Polygon :

Sans paramètres :

```
    editor.GetNew<Polygon>();
```

Avec paramètres :

```
    editor.GetNew<Polygon>(4,Point(0,0),2.f,Color::blue,Color::red,2);
    initializer_list<Point> l2={Point(0,0),Point(1,1),Point(2,2),Point(3,3)};
    editor.GetNew<Polygon>(l2,Color::green);
```

Cercle vide :

Sans paramètres :

```
    editor.GetNew<Circle>();
```

Avec paramètres :

```
    editor.GetNew<Circle>(Point(0,0),1.f);
    editor.GetNew<Circle>(Point(0,0),1.f,Color::red,2);
```

Disk :

Sans paramètres :

```
    editor.GetNew<Disk>();
```

Avec paramètres :

```
    editor.GetNew<Disk>(Point(0,0),1.f);
    editor.GetNew<Disk>(Point(0,0),1.f,Color::blue,Color::red,2);
```

Ellipse :

Sans paramètres :

```
    editor.GetNew<Ellipse>();
```

Avec paramètres :

```
    editor.GetNew<Ellipse>(Point(0,0),1.f,2.f);
    editor.GetNew<Ellipse>(Point(0,0),1.f,2.f,Color::green,Color::yellow,2);
```

Polyline :

Sans paramètres :

```
    editor.GetNew<Polyline>();
```

Avec paramètres :

```
    editor.GetNew<Polyline>(3,Point(0,0),2.f,Color::red,2);
    initializer_list<Point> l3={Point(0,0),Point(1,1),Point(2,2)};
    editor.GetNew<Polyline>(l3,Color::blue,2);
```

Toutes ces formes sont par la suite modifiables avec ces fonctions :

```
void Translate(const Point &v) { Array::Geometry::Translate(*this,v); }
void Scale(float s) { Array::Geometry::Scale(*this,s); }
void Rotate(float a) { Array::Geometry::Rotate(*this,a); }
void setFillColor(const Color &c) { fill_color = c; }
void setStrokeColor(const Color &c) { stroke_color = c; }
```

```
void setStrokeWidth(size_t w) { stroke_width = w; }
void setOpacity(float o) { opacity = o; }
```

Certaines formes ont d'autre fonctions permettant de modifier certain aspect de la forme.

Rectangle et Square:

```
void setWidth(float w) { width = w; }
void setHeight(float h) { height = h; }
```

Triangle :

```
void setPoints(const Point &p1, const Point &p2, const Point &p3) {
       (*this)[0] = p1; (*this)[1] = p2; (*this)[2] = p3;
}
```

Polyline et Polygon:

```
void setPoints(std::initializer_list<Point> list) {
Array::Fixed<Point>::operator=(list); }
```

Ellipse:

```
void setRadius(float x, float y) { rx = x; ry = y; }
void setCenter(const Point &c) { center = c; }
```

Circle et Disk :

```
void setRadius(float r) { radius = r; }
void setCenter(const Point &c) { center = c; }
```

Ajout de forme dans le canevas :

Pour ajouter une forme au canevas il faut utiliser la fonction Add() :

Le paramètre de la fonction est une forme :

```
    Polygon p1=editor.GetNew<Polygon>();
//On peut ensuite la modifier
    p1.Translate(Point(100,100));
//Puis l'ajouter
    editor.Add(p1);
```

Ou

```
    editor.Add(editor.GetNew<Polygon>());
```

(Cela n'a pas vraiment de sens de le faire comme ça car la forme par défaut n'est pas cohérente avec ce que l'on veut faire)

Toutes les formes précédemment citées peuvent être ajouter au canevas.

Annulation et reconstruction :

Pour annuler l'ajout d'une forme la fonction Undo() est disponible :

Elle prend en paramètre un entier qui indique le nombre d'annulation à effectuer, si aucun paramètre n'est entré la fonction effectue une annulation.

Pour annuler l'annulation ou annuler la suppression d'une forme la fonction Redo() est disponible :

Elle prend le même paramètre que la fonction undo.

Sélection de forme dans le canevas :

Pour sélectionner un ou plusieurs types de formes dans le canevas la fonction Select() est disponible :

Elle prend en paramètre un ou plusieurs membres de « enum shapetype » listés ici :

```
enum class ShapeType: uint64_t {
  Segment = 1ULL << 0,
  Triangle = 1ULL << 1,
  Carre = 1ULL << 2,
  Rectangle = 1ULL << 2,
  Polygone = 1ULL << 3,
  Pentagone = 1ULL << 3,
  Hexagone = 1ULL << 3,
  Heptagone = 1ULL << 3,
  Octogone = 1ULL << 3,
  Cercle = 1ULL << 4,
  Disk = 1ULL << 4,
  Ellipse = 1ULL << 5,
  Complexe = 1ULL << 6,
  Polyline = 1ULL << 6,
  All = 0xFFFFFFFFFFFFFFFFULL
};
```

S'il y a plusieurs membres il faut les séparer avec l'opérateur binaire |.

Par la suite, on pourra utiliser la fonction Print() afin d'afficher le contenu de la sélection. Si aucune forme n'a été sélectionné, le canevas affiche toutes les formes présentent.

Redimension du caneva :

Les fonctions Resize() et crop() permettent de redimensionner le canevas, elles prennent toutes deux en paramètre une largeur et une hauteur.

```
editor.Resize(3000,3000);
editor.Crop(800,800);
```

Si les dimensions données à la fonction Resize sont inférieurs à celle du canevas la fonction Crop est appelée.

Les formes qui se trouvent en dehors des dimensions sont supprimés et peuvent d'être restauré si la taille la permet grâce à la fonction Redo.

Export du canevas au format SVG :

La fonction ExportSVG() permet d'exporter les formes sélectionnées dans un fichier SVG.

La fonction prend en paramètre une chaine de caractère qui donnera le nom au fichier SVG.

Si aucune forme n'est sélectionnée la fonction sélectionne automatiquement toutes les formes.

Makefile :

Pour exécuter le makefile vous pouvez entrer la commande make run, cela compilera la bibliothèque et génèrera le fichier executable avant de le lancer automatiquement.

Sources utilisées :

Liste chainée : https://developpement-informatique.com/article/487/les-listes-chainees-en-c++ + cours INFO0401

Copilot pour la rédaction rapide et l'aide à la résolution de bug