TP4 C++

B3129 : Pierre-Louis LEFEBVRE et Nicolas SIX

Vendredi 5 février 2016

Table des matières

Première partie

Cahier des charges

1 Introduction

Notre logiciel est un éditeur de formes géométriques en ligne de commande (sans interface graphique). L'éditeur permet la gestion des formes géométriques suivantes dans un plan possédant un repère cartésien XY :

- Segment de droite
- Rectangle aux côtés parallèles aux axes du graphique
- Polygone convexe
- Réunion et intersection des formes précédentes et d'autres intersections et réunions

Toutes les formes sont considérées comme ń fermées ż c'est à dire que les points situés sur les bords de la formes font partie de la forme. Les objets peuvent être vides de tout point (les réunions et intersections notamment).

2 Commandes

2.1 Ajout d'une forme

Commande

COMMANDE Name PARAM

Réponse

[OK|ERR]

Ajoute une forme de type COMMANDE (voir ci-dessous) à partir des paramètres PARAM (voir ci-dessous) et de nom Name, qui est un mot composé de chiffres et de lettres majuscules ou minuscules uniquement. Les paramètres de type nombre sont toujours des entiers dans la limite de int du C++, sauf indication contraire. La réponse est OK si la commande s'est bien exécutée, ERR sinon. La réponse peut s'accompagner d'un commentaire : une ligne commençant par le caractère # indiquant dans le cas d'une erreur le type d'erreur (unkown command (lorsque COMMANDE n'est pas une commande du logiciel), name already used (lorsque le nom donné à l'objet à créer est déjà utilisé pour un autre objet), invalid parameters (lorsque les paramètres PARAM est invalide)).

2.2 Ajout d'un segment

Commande

S Name X1 Y1 X2 Y2

Ajoute un segment entre les points de coordonnées (X1, Y1), (X2, Y2), X1, Y1, X2, Y2 étant des nombres.

2.3 Ajout d'un rectangle

Commande

R Name X1 Y1 X2 Y2

Ajoute un rectangle dont le segment d'extrémités de coordonnées (X1, Y1) et (X2, Y2) est une diagonale.

2.4 Ajout d'un polygone convexe à n côtés

Commande

PC Name X1 Y1 X2 Y2 Xn Yn

Ajoute un polygone de côtés de coordonnées (X1, Y1), (X2, Y2) (Xn, Yn), avec n 3 uniquement si ce dernier est convexe, sinon une erreur sera générée avec le commentaire non convex polygon.

2.5 Ajout d'une réunion de N objets

Commande

OR Name Name1 Name2 NameN

Construit un objet Name comme la réunion des N objets Name1, Name2 NameN. N 0 (si N = 0 alors on crée un objet vide). Si le nom de l'un des N objets ne correspond pas à un objet déjà existant, une erreur de paramètres invalides est renvoyée.

2.6 Ajout d'une intersection de N objets

Commande

OI Name Name1 Name2 NameN

Construit un objet Name comme l'intersection des N objets Name1, Name2 NameN. N 0 (si N = 0 alors on crée un objet vide). Les erreurs possibles sont les mêmes que pour la réunion.

2.7 Suppression d'un objet

Commande

DELETE Name1 Name2 NameN

Réponse

[OK|ERR]

Supprime les N objets Name1, Name2 NameN. Si un nom ne correspond pas à un objet existant, aucun objet n'est supprimé et une erreur de paramètres invalides est renvoyée.

2.8 Déplacement d'un objet

Commande

MOVE Name dX dY

Réponse

[OK|ERR]

Déplace l'objet Name d'une distance dX (nombre) sur l'axe des X et dY (nombre) sur l'axe des Y

2.9 Vérification dappartenance d'un point à l'intérieur d'un objet

Commande

```
HIT Name X Y
```

Réponse

```
[YES|NO]
```

Renvoie YES si le point de coordonnées (X, Y) se trouve à l'intérieur (ou au bord) de l'objet Name NO sinon.

2.10 Énumération des formes géométriques existantes

Commande

```
LIST
```

Réponse

```
Desc1
Desc2
...
DescN
```

Affiche les descripteurs des formes géométriques existantes, selon le format suivant :

Pour le rectangle, le segment et le polygone, la description est identique à la commande d'ajout correspondante.

Exemple : Le rectangle de nom Name1 et de côtés opposés (2,3) et (3,8) :

```
R  Name1 2 3 3 8
```

Pour l'intersection et la réunion de N objets, la forme est la suivante. Les descripteurs complets de l'ensemble des N objets (y compris ceux d'éventuels autres intersections ou réunion), sauf que leur nom est le nom d'origine immédiatement suivi sans séparateur d'un tiret (-) et du nom de l'intersection ou de la réunion. Suivi de la commande de création de l'intersection ou de la réunion, et enfin de la commande de destruction des N objets.

Exemple : La réunion OR1 du rectangle R1, du segment S1 et de l'intersection OI1 du polygone P1 et du segment S2 :

```
R R1-OR1 1 2 3 8
S S1-OR1 1 7 45 48
PC P1-OI1-OR1 14 45 48 12 45 78 89 56 78 10
S S2-OI1-OR1 0 0 2 2
OI OI1-OR1 P1-OI1-OR1 S2-OI1-OR1
DELETE P1-OI1-OR1
```

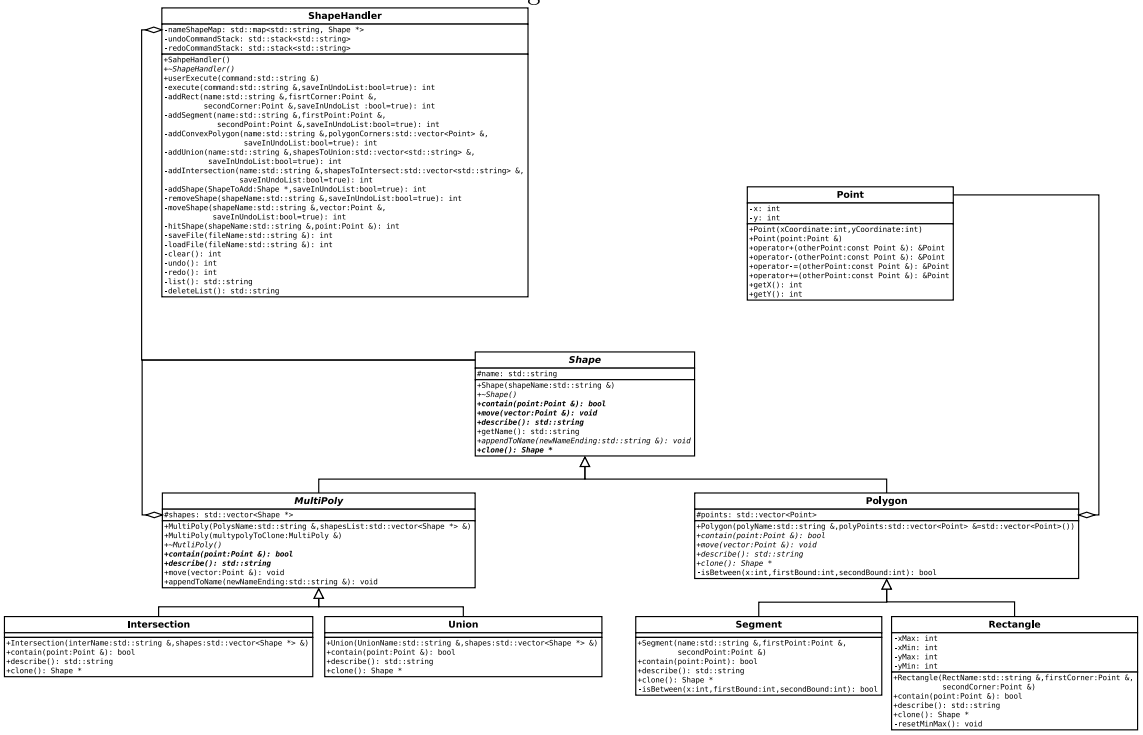
DELETE S2–OI1–OR1
OR OR1 R1–OR1 S1–OR1 OI1–OR1
DELETE R1–OR1
DELELTE S1–OR1
DELETE OI1–OR1

La persistance d'un ensemble d'objets construit de cette manière ; le fichier de sauvegarde est un fichier de format texte.

Deuxième partie
Conception

3 Diagramme UML

FIGURE 1 – Diagramme de classe UML



4 Détection des dappartenance d'un point à une forme

4.1 Appliquer à un ensemble

Chaque ensemble, intersections et réunions, contient chacun des éléments qui le compose. Pour effectuer la détection un ensemble se contente donc de vérifier si le point donné appartient ou non aux éléments qui le compose. Cette opération peut sembler coûteuse mais permet de ne pas avoir les problèmes dimprécision due aux arrondis qui seraient apparus lors du calcul d'une forme

qui résumerait cet ensemble. De plus il est facile de ne pas étudier chaque sous-forme dans les nombreux cas que représentent les réponses négatives.

Il apparaît donc qu'une détection efficace des hit sur les formes élémentaire est primordiale.

coût : $O(n)$ avec n le nombre de figure contenu dans l'ensemble.

4.2 Appliquer à un segment

La détection des hit sur un segment se base sur léquation de la droite correspondant au segment avec tout d'abord vérification de lappartenance au rectangle dont la diagonale est le segment considéré. Cette méthode est relativement complexe, car elle nécessite en plus de quelques additions une multiplication et surtout une division, toute fois ces calculs restent rapides et ne posent pas de problèmes de performance.

coût : $O(1)$

4.3 Appliquer à un rectangle

La détection des hit sur un rectangle et ici triviale vu qu'elle se compose de quelques comparaisons par rapport aux valeurs max et min de la figure suivant les deux axes une opération très rapide.

coût : $O(1)$

4.4 Appliquer à un Polygone convexe

La détection des hit sur un polygone convexe détermine si un point appartient à la figure en cherchant un segment se situant au-dessus du point et un en dessous. La grande majorité des cas sont traités en quelques conditions. Les cas les plus compliqués quant à eux utilisent le même système que pour les segments en utilisant l'équation du côté considéré. Cette méthode traite donc les côtés un par un juste à en trouver un au-dessus et un en dessous ou bien jusquà trouver un point appartenant à un côté. Bien que relativement lourds sur les polygones ayant de nombreux cotés, cette technique reste capable de traiter tous les côtés sauf dans le pire des cas deux d'entre eux juste en faisant quelques comparaisons de valeur ce qui lui permet de rester très efficace.

coût : $O(n)$ avec n le nombre de points du polygone.

5 Détection de la convexité d'un polygone

Afin de vérifier qu'un polygone est bien convexe, nous vérifions que le sinus de l'angle entre deux points formant un côté du polygone et un autre point du polygone reste toujours de même signe opération que nous répétons pour toutes les combinaisons possibles dans le polygone. Cette opération est assez lourde en calcul mais nous assure que le polygone et bien convexe et nous permet notamment de détecter les polygones croisés et repliées sur eux-mêmes.

coût : $O(n^2)$ avec n le nombre de points du polygone.