Compte-rendu du TP C++ 4
Cahier des charges détaillés

Arnaud Favier Marc Gagné

Février 2016

Préambule

L'application doit être un éditeur de formes géométriques permettant de manipuler ces formes simplement. L'application ne comportera pas d'interface graphique, et fonctionnera en mode console uniquement. L'interaction avec l'éditeur se fera en mode console aussi. L'éditeur contiendra un ensemble de formes géométriques appelé le modèle. L'éditeur doit permettre la gestion des formes géométriques (aussi appelées figures) suivantes :

- Le segment (défini par deux points)
- Le rectangle (défini par deux points : haut gauche et bas droit)
- Le polygone convexe (défini par une liste de points)

L'application doit également permettre d'effectuer des opérations au travers de commandes dédiées :

- L'ajout d'un nouvel objet au modèle actuel (avec l'une de formes présentées précédemment)
- La création d'un nouvel objet comme résultat d'une opération sur des objets existants. Les opérations pouvant être :
  - La réunion (regroupement dans un ensemble de un ou plusieurs objets dans leur totalité)
  - L'intersection (ensemble des parties communes des objets)
- La suppression d'un objet
- Le déplacement d'un objet (suivant une distance en abscisse et une en ordonnée)
- La vérification d'un point se trouvant à l'intérieur d'un objet ou non (exactement sur côté de l'objet est considéré comme intérieur)
- La persistance d'un ensemble d'objets construit de cette manière via :
  - Une sauvegarde du modèle courant (dans un fichier de sauvegarde au format texte)
  - Un chargement d'un modèle précédent (grâce au fichier de sauvegarde)

L'interaction avec cette application se fera en mode console uniquement. L'application reçoit des commandes en entrée. Une commande est définie par une chaîne des caractères terminée avec le symbole de fin de ligne `\n`. Après la réception d'une commande, l'application peut répondre avec un texte (qui peut être vide). L'application peut afficher à tout instant en console, des lignes des caractères qui commencent avec le caractère #. Ces lignes sont considérées comme des commentaires, et n'ont pas d'influence sur le fonctionnement des entrées/sorties.

Pour des raisons d'intégration avec d'autres modules déjà existants, l'application doit implémenter d'une manière très stricte l'interface en mode console, en respectant la syntaxe détaillée dans la section Commandes possibles.

1 Commandes possibles

Toutes les commandes d'ajout échouent si une figure existe déjà avec le même nom. De même, les commandes qui fournissent des noms de figure qui sont censées exister dans le modèle échouent si ces noms ne correspondent pas à des figures créées.

1.1 Ajouter un segment

Ajoute un objet de type segment entre le point de coordonnées (X1 ; Y1) et le point de coordonnées (X2 ; Y2). L'objet a un nom (`Name`), qui est une chaîne de caractère composée de lettres et/ou de chiffres, sans caractère séparateur (du type espace, `\n`, `\t`...) à l'intérieur. Les paramètres des coordonnées de la commande sont des entiers longs.

La réponse à la commande est `OK` si la commande s'est bien exécutée, `ERR` dans le cas contraire.

Commande	Réponse
`S Name X1 Y1 X2 Y2`	[`OK` | `ERR`]

Exemple :

```
C: S c11 12 15 45 20
R: OK
C: S 12 15
R: ERR
```

1.2 Ajouter un rectangle

Ajoute un objet de type rectangle défini par deux coordonnées de points : le point en haut à gauche de coordonnées (X1, Y1) et le point en bas à droite de coordonnées (X2, Y2). L'objet a un nom (`Name`), qui est une chaîne de caractère composée de lettres et/ou de chiffres, sans caractère séparateur (du type espace, `\n`, `\t`...) à l'intérieur. Les paramètres des coordonnées de la commande sont des entiers longs.

La réponse à la commande est `OK` si la commande s'est bien exécutée, `ERR` dans le cas contraire.

Commande	Réponse
`R Name X1 Y1 X2 Y2`	[`OK` | `ERR`]

Exemple :

```
C: R rectangle1 56 46 108 4536
R: OK
C: R 56 46
R: ERR
```

1.3 Ajouter un polygone convexe

Ajoute un objet de type polygone défini par une liste de coordonnées de points : (X1 ; Y1) ; (X2 ; Y2) ; (X3 ; Y3) ... (Xn ; Yn), avec n >= 3. L'objet a un nom (`Name`), qui est une chaîne de caractère composée de lettres et/ou de chiffres, sans caractère séparateur (du type espace, `\n`, `\t`...) à l'intérieur. Les paramètres des coordonnées de la commande sont des entiers longs. Il est rappelé qu'un polygone est convexe si l'ensemble de ses angles sont inférieurs à 180°. Il est supposé par le programme que le polygone fourni n'est pas croisé.

La réponse à la commande est `OK` si la commande s'est bien exécutée, `ERR` dans le cas contraire : si le polygone n'est pas convexe, une erreur sera générée (retourner `ERR`).

Commande	Réponse
`PC Name X1 Y1 X2 Y2 ... Xn Yn`	[`OK` | `ERR`]

Exemple :

```
C: PC Name12 56 46 108 4536 80 100
R: OK
C: PC 56 46
R: ERR
```

1.4 Opération de réunion

Permet de construire un nouveau objet appelé Name comme la réunion de la liste d'objets existants énumérés par leur nom : `NameObj1`... `NameObjN` (de 1 à N). La réunion est un regroupement dans un ensemble de un ou plusieurs objets dans leur totalité. Si au moins un des noms fourni ne correspond pas à une figure existant dans le modèle, la création échoue et la commande renvoie `ERR`.

Commande	Réponse
`OR Name NameObj1 NameObj2... NameObjN`	[OK | ERR]

Exemple :

```
C: PC Name12 56 46 108 4536 80 100
R: OK
C: S seg1 56 46 108 4536
R: OK
C: OR re1 seg1 Name12
R: OK
C: OR fig0
R: ERR
```

1.5 Opération d'intersection

Permet de construire un nouveau objet appelé Name comme l'intersection de la liste d'objets existants énumérés par leur nom : `NameObj1`... `NameObjN` (de 1 à N). L'intersection est l'ensemble des parties communes des objets. Si au moins un des noms fourni ne correspond pas à une figure existant dans le modèle, la création échoue et la commande renvoie `ERR`.

Commande	Réponse
`OI Name NameObj1 NameObj2... NameObjN`	[OK | ERR]

Exemple :

```
C: PC Name12 56 46 108 4536 80 100
R: OK
C: S seg1 56 46 108 4536
R: OK
C: OI int 1 seg1 Name12
R: OK
C: OI fig0
R: ERR
```

1.6 Opération d'appartenance

Permet de vérifier si le point de coordonnées (X ; Y) se trouve bien à l'intérieur de l'objet appelé `Name`. Si le point se trouve exactement sur le bord de l'objet, on le considérera comme à l'intérieur de ce dernier.

Commande	Réponse
`HIT Name X Y`	[YES | NO]

Exemple :

```
C: R rectangle1 56 46 108 4536
R: OK
C: HIT rectangle1 57 50
R: YES
C: HIT rectangle1 56 46
R: YES
C: HIT rectangle1 52 38
R: NO
```

1.7 Suppression

Supprime les objets existants énumérés par leur nom : `NameObj1`... `NameObjN` (de 1 à N) qui sont passés en paramètre. Si un nom est invalide, aucun objet n'est supprimé, et une erreur `ERR` est renvoyée.

Commande	Réponse
`DELETE NameObj1 NameObj2... NameObjN`	[OK | ERR]

Exemple :

```
C: DELETE rectangle1 Name1
R: OK
```

1.8 Déplacement

Déplace l'objet appelé `Name` d'une distance de dX sur l'axe des abscisses (x) et d'une distance de dY sur l'axe des ordonnées (y).

Commande	Réponse
`MOVE Name dX dY`	[OK | ERR]

Exemple :

```
C: MOVE Name12 100 -25
R: OK
```

1.9 Énumération

Affiche les descripteurs des objets existants, dans l'ordre inverse de l'ordre de création. Si aucun objet n'est défini, la commande n'affiche rien. Le format d'affichage est celui défini dans la section Format d'affichage et de fichier. La liste est triée alphabétiquement.

Commande	Réponse
LIST	[Desc1 Desc2 ... DescN | (vide)]

LIST [Desc1 Desc2 ... DescN | (vide)]

Exemple :

```
C: LIST
R : Name12 C [(56;46) (108;4536) (80;100)]
R : re1 U [S [(56;46) (108;4536)]  C [(56;46) (108;4536) (80;100)]]
R : seg1 S [(56;46) (108;4536)]
```

1.10 Annuler la dernière opération

Annuler la dernière opération qui a eu un effet sur le modèle : insertion d'un objet, suppression, déplacement, chargement d'un fichier... Si le début de l'historique a été atteint, la commande renvoie `ERR`.

Commande	Réponse
UNDO	[OK | ERR]

Exemple :

```
C: UNDO
R: OK
```

1.11 Refaire la dernière opération

Ré-exécuter la dernière opération annulée qui a eu un effet sur le modèle. La commande `REDO` a un effet sur le modèle s'il y a eu au moins une commande `UNDO` précédemment ; sinon, la commande renvoie `ERR`.

Commande	Réponse
`REDO`	[`OK` | `ERR`]

Exemple :

```
C: REDO
R: OK
```

1.12 Sauvegarder le modèle courant

Sauvegarde le modèle courant (l'ensemble des formes géométriques courantes) dans un fichier au format texte à la racine du dossier contenant le projet. Le format du fichier est celui défini dans la section Format d'affichage et de fichier.

Commande	Réponse
`SAVE Filename`	[`OK` | `ERR`]

Exemple :

```
C: SAVE a.txt
R: OK
```

1.13 Charger un modèle en mémoire

Charge un ensemble d'objets dans le modèle courant à partir d'un fichier au format texte. Le format du fichier est celui défini dans la section Format d'affichage et de fichier. Si au moins une des figures du modèle à charger a le même nom qu'une figure dans le modèle existant, le chargement est annulé, aucune modification du modèle en cours n'est effectuée et la commande renvoie `ERR`.

Commande	Réponse
`LOAD filename`	[`OK` | `ERR`]

Exemple :

```
C: LOAD a.txt
R: OK
```

1.14 Vider le modèle courant

Supprime tous les objets composant le modèle actuel. Le modèle étant l'ensemble des formes géométriques courantes de l'application.

Commande	Réponse
`CLEAR`	[`OK` | `ERR`]

Exemple :

```
C: CLEAR
R: OK
```

1.15 Fermer l'application

Quitte l'application sans afficher de message.

Commande	Réponse
EXIT	(aucune)

Exemple :

```
C: EXIT
R:
```

2 Format d'affichage et de fichier

Lors des différentes commandes de l'application, il est parfois nécessaire d'afficher les formes géométriques en console, ou de les enregistrer dans un fichier au format texte. Pour cela, le format suivant sera adopté :

`Name Type [(Point1X;Point1Y) (Point2X;Point2Y) ... (PointNX;PointNY)]` pour les formes simples (formes géométriques), ou `Name Type [Type [(Point1X;Point1Y) (Point2X;Point2Y)] Type [(Point1X;Point1Y) (Point2X;Point2Y)]]` pour les objets plus complexes (unions, intersections).

Exemple :

```
C: PC Name12 56 46 108 4536 80 100
R: OK
C: S seg1 56 46 108 4536
R: OK
C: OR re1 seg1 Name12
R: OK
C: SAVE test
R: OK
C: LIST
R: Name12 C [(56;46) (108;4536) (80;100)]
R: re1 U [S [(56;46) (108;4536)]  C [(56;46) (108;4536) (80;100)]]
R: seg1 S [(56;46) (108;4536)]
C: SAVE test.txt
R: OK
```

Contenu de `test.txt` :

```
Name12 C [(56;46) (108;4536) (80;100)]
re1 U [S [(56;46) (108;4536)]  C [(56;46) (108;4536) (80;100)]]
seg1 S [(56;46) (108;4536)]
```