**TP - Héritage & Polymorphisme**

**Cahier des charges**

Implémenter un éditeur de formes géométriques, créations et manipulations de celles-ci. La réalisation de l’interface graphique n’est pas demandé, l’interaction avec l’éditeur ce fera uniquement en mode console.

L’éditeur doit permettre la gestion des formes suivantes :

* Segment
* Rectangle
* Polygone convexe

Et doit également pouvoir effectuer ces différentes opérations :

* Ajout d’un nouvel objet (d’une des formes précédentes)
* Création d’un nouvel objet issu d’une réunion ou d’une intersection de plusieurs objets déjà existant.
* Suppression d’un objet
* Déplacement d’un objet
* La vérification de l’appartenance d’un point à un objet
* La sauvegarde de l’état actuel (les objets et leurs caractéristiques) dans un fichier format texte
* Le chargement d’une sauvegarde

L’interaction avec l’application se fait en mode console, elle reçoit en entrée des commandes (des chaînes de caractères terminées par le symbole de fin de ligne « \n »). A la réception de cette commande, l’application a la possibilité de répondre un texte (qui peut être vide). De même elle peut afficher dans la console à tout instant des lignes de caractères commençant par un « # », ces lignes sont considérées comme des commentaires et n’influencent en aucun cas le fonctionnement des entrées/sorties.

On considérera que la syntaxe des commandes est correcte, c’est-à-dire que tous les paramètres nécessaires sont renseignés par l’utilisateur.

Commandes possibles

Ajouter un segment

Commande :

S Name X1 Y1 X2 Y2

Réponse :

OK | ERR

| # Points identiques

Description : Ajoute un segment entre les points (X1,Y1) et (X2,Y2). Ce segment possède un nom (Name) composé de lettres et/ou de chiffres, aucun séparateur ou caractère spécial ne devra être renseigné. Ces valeurs (X1, Y1, X2, Y2) sont des entiers, positifs ou négatifs. Dans la suite du cahier des charges, tous les paramètres sont des entiers, sauf mention contraire explicite. Si les deux points en entrées sont les mêmes, une erreur sera générée.

La réponse de l’application est OK si la commande s’est bien exécutée. Dans le cas contraire ERR est affiché avec à la ligne un «# » suivi du motif de l’erreur.

Exemple :

C : S unSegment1 -3 5 1 1

R : OK

Exemple 2 :

C : S unSegment1 1 1 1 1

R : ERR

R : # Points identiques

Ajouter un rectangle

Commande :

R Name X1 Y1 X2 Y2

Réponse :

OK | ERR

| # Mauvais cadran

Description : Ajoute un rectangle défini par les deux points (X1, Y1) et (X2, Y2) qui sont respectivement le point en haut à gauche et le point en bas à droite du rectangle.

Si les deux points entrés sont identiques ou s’ils ne sont pas respectivement à leur place (X2 < X1 et/ou Y2 > Y1) une erreur est générée.

Exemple :

C : R unRectangle1 1 3 6 0

(X1, Y1)

(X2, Y2)

R : OK

Exemple 2 :

C : R unRectangle1 6 0 1 3

R : ERR

R : # Mauvais cadran

Figure

Ajouter un polygone convexe

Commande :

PC Name X1 Y1 X2 Y2 …. Xn Yn

Réponse :

OK | ERR | ERR

| # Polygone non convexe | # Nombre de points insuffisants

Description : Ajoute un polygone défini par les points (X1, Y1), (X2, Y2),(X3,Y3), … , (Xn, Yn) avec n ≥ 3. Si n < 3, une erreur sera générée au même titre que si le polygone est croisé ou n’est pas convexe.

(X1 , Y1)

(X2 , Y2)

(X3 , Y3)

(X4 , Y4)

(X5 , Y5)

Exemple :

C : PC PolyConvex1 1 1 3 -5 -2 -1

R : OK

Exemple 2 :

C : PC PolyConvex1 4 4 3 0 7 3 3 -3 0 0

R : ERR

R : # Polygone non convexe

Opération de réunion

Figure

Commande :

OR Name Name1 Name2 … NameN

Réponse :

OK | ERR |ERR

| # Pas de forme pour faire un ensemble |# $currentName inexistant

Description : Permet de construire un nouvel objet Name issu de la réunion de la liste d’objets déjà existants (Name1, Name2, Name3 … NameN). Si aucun nom n’est donné ou que l’un est erroné une erreur est générée.

Exemple :

C : PC PolyConvex1 1 1 3 -5 -2 -1

R : OK

C : R Rectangle1 1 3 6 0

R : OK

C : OR Reunion1 PolyConvex1 Rectangle1

R : OK

Opération d’intersection

Commande :

OI Name Name1 Name2 … NameN

Réponse :

OK | ERR |ERR

| # Pas de forme pour faire un ensemble |# $currentName inexistant

Description : Idem que réunion, mais le nouvel objet est une intersection.

Exemple :

C : PC PolyConvex1 1 1 3 -5 -2 -1

R : OK

C : R Rectangle1 1 3 6 0

R : OK

C : OI Reunion1 PolyConvex1 Rectangle1

R : OK

Appartenance

Commande :

HIT Name X Y

Réponse :

YES | NO

Description : Permet de vérifier que le point (X, Y) appartient à l’objet Name. On dira qu’un point appartient à un objet s’il n’est ni à l’extérieur ni sur le bord de celui-ci.

Exemple :

C : R Rectangle1 1 3 6 0

R : OK

C : HIT Rectangle1 3 2

R : OK

Suppression – Vider le modèle

Commande :

DELETE Name1 Name2 … NameN

Réponse :

OK | ERR

| # Nom inexistant

Description : Supprime les objets listés. Si un nom est invalide, aucun objet n’est supprimé et une erreur est renvoyée.

Exemple :

C : DELETE Rectangle1 Segment1

R : OK

2nd Commande :

CLEAR

Réponse : OK

Description : Supprime tous les objets que contient le modèle actuel.

Exemple :

C : CLEAR

R : OK

Déplacement

Commande :

MOVE Name dX dY

Réponse :

OK | ERR

| # Nom inexistant

Description : Permet de déplacer l’objet Name de dX sur l’axe des abscisses et de dY sur l’axe des ordonnées.

Exemple :

C : MOVE Rectangle1 8 -12

R : OK

Enumération

Commande :

LIST

Réponse :

DescrName1

DescrName2

…

DescrNameN

Description : Affiche les descripteurs des objets composant le modèle actuel. Pour le segment, le rectangle et le polygone convexe nous conserverons la syntaxe des commandes. Quant aux réunions et intersections nous utiliserons une syntaxe particulière, nous privilégierons la capacité à enregistrer et charger le modèle à une syntaxe claire.

Exemple :

C : LIST

R : R Rectangle1 1 3 6 0

R : Rectangle2 1 3 6 0

R : R \_Rectangle1 1 3 6 0

R : R \_Rectangle2 1 3 6 0

R : OR Reunion1 \_Rectangle1 \_Rectangle2

R : DELETE \_Rectangle1 \_Rectangle2

Annuler la dernière opération

Commande :

UNDO

Réponse :

OK | ERR

| # (Pas d’ancienne opération)

Description : Annule la dernière opération de l’utilisateur qui a eu un effet sur le modèle, tel que le déplacement, la suppression, l’insert d’objet et le chargement de fichier. Si aucune de ces opérations n’a été effectuée avant un UNDO, une erreur est générée. L’application ne pourra annuler uniquement les 20 dernières opérations.

Exemple :

C : UNDO

R : OK

Reprendre la dernière modification

Commande :

REDO

Réponse :

OK| ERR | ERR

| # (Pas d’ancienne opération) | #Commande entre le undo et le redo.

Description : Refaire la dernière opération annulée (avec UNDO) qui a eu un effet sur le modèle. REDO fonctionne uniquement s’il n’y a pas eu d’opération qui a eu un effet sur le modèle entre le dernier UNDO et le REDO. Dans le cas contraire une erreur est levée. De même, REDO ne peut revenir qu’aux 20 dernières opérations.

Exemple :

C : REDO

R : OK

Charger en mémoire un modèle

Commande :

LOAD filename

Réponse :

OK | ERR

| # (Fichier inexistant)

Description : Charge un ensemble d’objet (un modèle) à partir d’un fichier texte (.txt).

Exemple :

C : LOAD fichier.txt

R : OK

Sauvegarder un modèle courant

Commande :

SAVE filename

Réponse : OK

Description : Sauvegarde les descripteurs du modèle actuel dans un fichier .txt. Si le fichier n’existe pas, il est créé. Si un fichier existe déjà avec le même nom, il est remplacé par le nouveau.

Fermer l’application

Commande :

EXIT

Description : Ferme l’application.