INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

MEYER Franck
ROYER Robin
B3231

TP 4 C++ : Héritage

Index

Spécifications

Introduction

Le TP a pour but la réalisation d'une application permettant de créer des formes géométriques simples (Cercle, Rectangle, Ligne, PolyLigne), ainsi que de faire des opérations dessus (déplacement, sélection, effacement, sauvegarde, ...)

Première partie

Généralités

1. Choix généraux

Commandes :

Le contrôle de l'éditeur de formes géométriques par l'utilisateur se fera à l'aide d'un shell interactif

Il y a 2 types de commandes :

- Les commandes qui agissent sur le modèle, en modifiant les objets le composant
- Les commandes de gestion de modèle: permettant de le visualiser ou de le sauvegarder

Les lignes commençant par le caractère **#** sont des commentaires et ne sont pas pris en compte.

Usage	Description
C name xc yc radius	Crée un cercle de centre (xc,yc) et de rayon (radius)
R name x1 y1 x2 y2	Crée un rectangle défini par 2 points : 1 et 2
L name x1 y1 x2 y2	Crée une ligne définie par 2 points : 1 et 2
PL name x1 y1 ... xn yn	Crée un polyligne défini par n points : 1 … n
S name x1 y1 x2 y2	Crée une séléction définie par 2 points : 1 et 2
DELETE name1 … nameN	Efface les formes nommées name1, …, nameN
MOVE name dx dy	Déplace name de (dx,dy)
LIST	Affiche la description de tous les objets, dans l'ordre alphabétique
UNDO	Annule la dernière action
REDO	Réapplique la dernière action annulée
LOAD filename	Charge en mémoire le fichier filename
SAVE filename	Sauvegarde le modèle dans le fichier filename
CLEAR	Efface tous les objets du modèle
EXIT	Arrête l'application

Parcours du fichier de sauvegarde : Le fichier de log est parcouru. Si une commande n'est pas acceptée[1], aucune ne sera exécutée (donc rien ne sera chargé).
Les commandes acceptées sont : C, R, L, PL

Structure de données(stockage du modèle) : Les objets sont divisés en 2 éléments principaux : les Formes, physiquement présentes dans le modèle, et les Sélections.

Structure de données(historique) : Pour l'historique (UNDO/REDO), nous avons adopté un design pattern, avec pour base une classe « History », possédant des classes filles dédiées au traitement de l'historique de chaque commande.

1 Hors commentaire (débutant par #)

Deuxième partie

Spécifications des méthodes

1. Fichier main.cpp

main :

Fonction d'entrée de l'application. Traite les commandes entrées à partir du shell interactif mis en place.

2. Fichier object.cpp

Constructeur :

Paramètre d'entrée:

- **name** : nom de l'objet créé

getName :

Méthode permettant de connaître le nom d'un objet.

move :

Méthode permettant de déplacer un objet

Paramètre d'entrée :

- **dx** : Le déplacement à appliquer à l'objet sur l'axe x
- **dy** : Le déplacement à appliquer à l'objet sur l'axe y

isSel:

Méthode permettant de différencier les objets de type Sélection des autres

Retour :

- **true** si l'objet est une sélection, false dans tous les autres cas

3. Fichier shape.cpp

Constructeur :

Paramètre d'entrée:

- **name** : nom de l'objet créé

getDescriptor :

Méthode permettant de renvoyer le descripteur d'une forme.

Sortie :

- Une chaîne de caractères formatée, représentant la commande nécessaire à créer cette forme

remove :

Méthode permettant de supprimer virtuellement une forme

remake :

Méthode permettant de recréer virtuellement une forme

isEncompassed :

Méthode permettant de savoir si une forme est à l'intérieur d'une séléction rectangulaire

Paramètres d'entrée :
- **x1**, **y1**, **x2**, **y2** : Les coordonnées définissant le rectangle de sélection

Sortie :
- **true** si ell l'est, **false** sinon

isDeleted :

Méthode permettant de savoir si une forme est virtuellement supprimée

Sortie :
- **true** si elle l'est, **false** sinon

4. Fichier selection.cpp

Constructeur :

Paramètres d'entrée:
- **name** : nom de l'objet créé
- **objects** : la liste des formes étant englobées par la sélection

remove :

Méthode permettant de supprimer une sélection, et de revoyer toutes les formes concernées par cette suppression

Sortie:
- La liste des formes supprimées en même temps que la sélection

5. Fichier objectmanager.cpp

addObject :

Méthode permettant d'ajouter une forme au gestionnaire

Paramètres d'entrée:
- **s** : La forme à ajouter
- **archivate** : booléen. Vaut true si l'on doit enregistrer cette opération dans l'historique

addSelection :

Méthode permettant d'ajouter une sélection au gestionnaire

Paramètres d'entrée:
- **s** : La sélection à ajouter

getSelectedShapes :

Méthode permettant de récupérer toutes les formes englobées par la sélection

Paramètres d'entrée:

- **x1, y1, x2, y2** : Les coordonnées englobant la sélection

contains :

Méthode permettant de vérifier si un objet du même nom que celui demandé est déjà présent

Paramètres d'entrée:

- **name** : Le nom à vérifier

Sortie:

- **true** si **name** est déjà présent

containsNames :

Méthode permettant de vérifier si tous les noms passés en paramètre sont déjà présent

Paramètres d'entrée:

- **names** : Les noms à vérifier

Sortie:

- **false** si au moins un nom n'est pas présent

removeObjects :

Méthode permettant de supprimer tous les objets dont le nom est donné

Paramètres d'entrée:

- **names** : Les noms des objets à supprimer
- **archivate** : booléen. Vaut true si l'on doit enregistrer cette opération dans l'historique

moveObjects :

Méthode permettant de déplacer l'objet dont le nom est donné

Paramètres d'entrée:

- **name** : Les noms del'objet à déplacer
- **dx**, **dy** : Le déplacement à appliuqer à cet objet
- **archivate** : booléen. Vaut true si l'on doit enregistrer cette opération dans l'historique

printDescs :

Méthode permettant d'afficher la liste des descriptions de tous les objets

Paramètres d'entrée:

- **os** : Le flux sortant où doit être affichée la liste

clearObjects :

Méthode permettant de supprimer tous les objets courants

Paramètres d'entrée:

- **archivate** : booléen. Vaut true si l'on doit enregistrer cette opération dans l'historique

addToHistory :

Méthode permettant d'ajouter un nouvel élément à l'historique

Paramètres d'entrée:

- **h** : l'élément à ajouter

undoCurrent :

Méthode permettant d'annuler la dernière opération

redoCurrent :

Méthode permettant de refaire la dernière opération

6. Fichier history.cpp

undo :

Méthode permettant d'annuler l'opération à laquelle est associée l'élément

Paramètres d'entrée:

- **om** : Pointeur vers le gestionnaire d'objets

redo :

Méthode permettant de refaire l'opération à laquelle est associée l'élément

Paramètres d'entrée:

- **om** : Pointeur vers le gestionnaire d'objets

Troisième partie

Diagramme de classes