Arnaud Dupeyrat
Antoine Breton
3IF - TP4 - CDC

CAHIER DES CHARGES DETAILLE

TABLE DES MATIERES

I Introduction

Le rôle de notre application est de gérer une suite d'actions prédéfinies par l'utilisateur.

Ces actions portent sur un ensemble de figures géométriques (Rectangle, Segment, Polygone convexe) qui forment un **modèle** qui pourra être étendu dans le futur.

Les actions supportées par l'application permettent de modifier ou de vérifier le contenu de notre modèle et s'expriment sous forme de plusieurs requêtes.

II Fonctionnalites

Voici les fonctionnalités accessibles par l'utilisateur sous formes de **commandes** :

1. La création d'une figure géométrique (Rectangle, Segment, Polygone). **Commande = 'R', 'S', 'PC'**
2. La suppression d'une figure géométrique. **Commande = 'DELETE'**
3. Une opération de réunion, permettant la construction d'un nouvel objet. **Commande = 'OR'**
4. Une opération d'intersection, permettant la construction d'un nouvel objet. **Commande = 'OI'**
5. La possibilité de tester l'appartenance d'un point vis à vis d'une figure. **Commande ='HIT'**
6. La possibilité de déplacer une figure. **Commande = 'MOVE'**
7. La possibilité d'afficher le contenu de notre application. **Commande = 'LIST'**
8. Une opération d'annulation. **Commande = 'UNDO'**
9. Une opération pour effectuer l'opération précédemment annulée. **Commande = 'REDO'**
10. Une opération permettant de sauvegarde la liste de commande depuis le début. **Commande = 'SAVE'**
11. Une opération permettant d'externaliser l'état du modèle a un instant t, dans un fichier lisible par l'humain. **Commande = 'SAVEC'**
12. Une opération permettant de charger le modèle à partir d'une sauvegarde précédemment effectuée. **Commande = 'LOAD'**
13. Une opération permettant de vider le modèle actuel. **Commande = 'CLEAR'**
14. Une opération permettant d'afficher la liste d'actions. **Commande = 'DISPACTION'**
15. Une commande permettant de quitter le programme. **Commande = 'EXIT'**

III Implementation

Pour répondre au besoin fonctionnel du cahier des charges initial, nous avons ajouté des contraintes techniques d'implémentation pour notre application. La principale motivation de ce choix étant d'améliorer la maintenabilité et la réutilisabilité de cette application.

Ainsi la conception devra répondre aux critères suivants :

- Optimisation de la réutilisabilité grâce au polymorphisme et aux design patterns.
- Performance importante afin d'être le plus efficace possible lors les traitements des commandes.

IV Remarques

Ce cahier des charges est construit en adéquation avec le cahier des charges initial fournis durant le TP.

Afin de ne pas surcharger le document nous ne reprendrons seulement les points qui nécessitent une explication approfondie ou les points non traités dans le sujet du TP, ainsi que les ajouts de fonctionnalité.

Dans le même ordre d'idée les « figures classiques » représentent le rectangle, le segment et le polygone convexe.

Ce qui concerne de près l'implémentation (design patterns, architecture, etc) est détaillé dans le document "spécification" et non dans celui ci.

V COMMANDES LIEES A LA MODIFICATION DU MODELE

1) LA CREATION D'UNE FIGURE.

Notre application permet la création informatique de figures géométriques, celles-ci reposent sur des points de type (X,Y).

Il nous semblait nécessaire qu'à chaque ajout de nouvelle figure notre programme teste la validité de la requête de l'utilisateur. En effet, une figure qui ne respecterait pas les conditions décrites ci-dessous n'est pas intégrée au modèle.

Après avoir averti l'utilisateur (message 'ERR'), l'application se chargera de traiter la requête suivante.

1. - La création d'un rectangle devra être associée à deux points.
2. - La création d'un segment devra être associée à deux points.
3. - La création d'un polygone devra être associée à trois (ou plus) points

De plus, nous laissons la possibilité à l'utilisateur d'effectuer des requêtes contenant des points identiques. Lors des différents algorithmes, les points confondus seront alors traités comme des points uniques. (angles, distances, ect).

Si tous les critères sont respectés alors la figure est créée et ajoutée au modèle.

Après avoir prévenu l'utilisateur, (message 'OK') le programme traite la prochaine commande.

Concernant l'insertion et la création d'une nouvelle figure il est nécessaire d'éclaircir quelques points du cahier des charges initial:

- Premièrement, notre programme offre une certaine flexibilité. En effet, l'application à la possibilité de traiter des points de types flottants. Ce qui nous permet d'élargir l'ensemble des possibilités offertes par notre application.
- Deuxièmement, nous avons fait le choix concernant le polygone convexe de l'ajouter au modèle si et seulement si la figure est un polygone strictement convexe.
- Troisièmement, le nom désignant la figure est unique. Si l'utilisateur décide d'ajouter deux figures comportant le même nom, l'insertion est autorisée mais supprime alors l'ancienne figure avec le même nom.
- Dernièrement, nous pouvons alors considérer le nom comme étant un identifiant unique et obligatoire de la figure. (Tout ajout de figure sans nom retournera un message d'erreur 'ERR').

2) Creation d'un composite (reunion, intersection).

Notre application permet la création informatique de figures géométriques décrivant l'intersection ou la réunion de figures.

Afin de créer le composite, l'utilisateur fait référence aux figures à l'aide d'un ensemble de noms, nous acceptons la création du composite dès lors que 2 noms (au minimum référence des figures réellement existantes dans le modèle) ; Le programme avertit l'utilisateur (message 'ERR') puis l'application se chargera de traiter la requête suivante.

Dans le cas contraire, après avoir averti l'utilisateur (message 'OK'), le programme peut alors créer le nouveau composite et l'insérer dans le modèle actuel.

Une fois inséré, il est alors possible d'effectuer l'ensemble des opérations possibles sur des figures « classiques » sur le composite.

3) Deplacer figure.

Afin de détailler le cahier des charges initial, il est important d'éclaircir quelques points :

- Si le nom de la figure recherchée par l'utilisateur n'est pas présent au sein du modèle alors le programme n'effectue aucune action. Après avoir averti l'utilisateur (message 'ERR') le programme traite la ligne de commande suivante.

4) Tester l'appartenance d'un point.

Afin de détailler le cahier des charges initial, il est important d'éclaircir quelques points :

- Premièrement, si le nom de la figure recherchée par l'utilisateur n'est pas présent au sein du modèle alors le programme n'effectue aucune action. Après avoir averti l'utilisateur le programme traite la ligne de commande suivante.
- Deuxièmement, nous avons fait le choix de considérer comme interne les points appartenant aux segments de la figure.

.

5) Affichage du modele.

A travers l'affichage, nous obtenons une vue d'ensemble des figures classiques ou composites qui composent notre modèle à un instant donné.

L'affichage du modèle est différent en fonction de figure traitée :

-> Premièrement, l'affichage des figures classiques sera sous le format suivant:

« Type Nom Point1 … PointN»

- **Type** : '**R**' = rectangle, '**S**' = Segment, '**PC**' = polygone convexe.
- **Nom:** représente l'identifiant unique de la figure, son nom
- **Point1 … PointN** : représente l'ensemble des points qui composent la figure.

- >Deuxièmement, l'affichage des composites (réunion ou intersection de figures aura le format suivant :

nom type

{

Type Nom Point1 … PointN

Type Nom Point1 … PointN

}

- **Type** : Permet de différencier une intersection d'une réunion
- **Le corps** : Représente l'ensemble des figures qui ont servi à construire le composite.

IV AUTRE FONCTIONNALITES

1) INTERACTION AVEC L'UTILISATEUR.

L'ensemble des interactions avec l'utilisateur se fait en ligne de commandes via des chaines de caractères suivant la forme suivante (commande + chaînes de caractères).

Si la commande n'est pas reconnue parmi l'ensemble des commandes possibles citées dans l'introduction, le programme n'effectuera aucune action. Après avoir averti l'utilisateur, le programme traite la ligne de commande suivante.

Il sera nécessaire de noter que les commandes sont en majuscules et que toute commande en minuscule ne sera pas traitée par l'application.

2) UNDO\REDO.

Afin de détailler le cahier des charges initial, il est important d'éclaircir quelques points :

- Nous laissons la possibilité de faire UNDO si et seulement si une action a déjà été réalisée.
- Nous laissons la possibilité de faire des REDO si et seulement si il y a eu un UNDO auparavant (Il est évident que le nombre de REDO possible dépend du nombre de UNDO effectué).
- Dés lors qu'un UNDO est effectué et qu'une action modifie le modèle, il est alors impossible d'effectuer un REDO.

L'ensemble des actions associées aux commandes décrites dans l'introduction ont la possibilité de UNDO\REDO si les caractéristiques cités ci-dessus sont respectées.

3) Sauver et charger le modele.

En réponse au cahier des charges initial, il est important d'éclaircir quelques points. Ainsi que de trouver un équilibre entre difficulté technique, temps et respect des fonctionnalités.

Dans cette partie nous allons détailler cet équilibre.

Les contraintes que nous sommes fixé sont les suivantes :

- le système de sauvegarde doit permettre a un **humain de visualiser** l'état du modèle a un instant t,
- le système doit permettre de **créer plusieurs fichiers** de sauvegarde,
- le système doit permettre **d'enregistrer** les différentes figures puis de les **recharger**.

Pour composer avec les contraintes fonctionnelles définies ci-dessus, les contraintes de temps ainsi que les difficultés technique nous proposons la solution suivante.

Nous fournissons deux commandes:

- **SAVE** <nomFic>, la fonction SAVE externalise dans un fichier, la liste des commandes effectuées depuis le début.
- **LOAD** <nomFic>, ouvre le fichier fourni en paramètre puis injecte les commandes qu'il contient dans le parser.
- **SAVEC** <nomFic>, enregistre le modèle à un instant t, les figures composites sont représentées grâce à des balises et les figures primaires sont représenté par la commande qu'il faudrait saisir pour créer cette figure à l'instant t.